

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

an Avid Media Group, Inc. newsletter

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

LightWave
5

Inside:

Get a Look at 3.5

GRAPH News

See the
Babylon 5 Way,
Part I



THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

Modeler
Mania

Inside:

Effects de 3D Art

Line Patching Pitfalls

Understanding Macros

What's on the
LightWavePRO
Disk?
See Page 18

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

Destroy
Your Own
Spaceship

Inside:

- Land Vehicle Movement
- Rendering Algorithms—Part II
- You Bonehead!
A Beginner's Look at Bones

What's on the
LIGHTWAVEPRO
Disk?
See Page 13

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

Understanding
Refraction

Inside:

A Collection of
Tips and Tricks
The Complete
Nebula Guide
Metaform Magic

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

Burning Up
The Screen

Inside:

Riding The Rails
Splitting Hairs in
LightWave
Flying Through
Canyons

THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

A First Look at
Metaform

What's on the
LIGHTWAVEPRO
Disk?
See Page 18

Inside:

Space the
Babylon 5 Way,
Part II

Hair-Raising Effects
Faking Retraction

Special 32 page LightWave Pro
supplement — exclusive to

AMIGA
COMPUTING



EDITOR'S MESSAGE

by John Gross

I became addicted to LightWave 3D when I first saw it being demonstrated by Allen Hastings at a Chicago Ami-Expo in 1990. Its intuitive interface and incredible picture quality made me jump off of the Turbo Silver wagon pretty quickly. Landing on the ground, I immediately rebounded up into the LightWave bandwagon. The first day I installed my Video Toaster, I remember staying up all night playing with LightWave. I didn't stop until the sun rose.

A few years later, I am still addicted to LightWave, so much so that it is now a major part of my life and I use it every day in my line of work. I've seen four versions of this program come along and each one keeps getting better and better. Now LightWave is available for many different computer platforms, and I want to see it being used everywhere!

About a year and a half ago, we came up with the idea of LIGHTWAVEPRO. Its purpose is to supply the LightWave community with a source of professional tips, tricks and tutorials. We've received a great response and are continuing to push the "LightWave envelope" with each issue.

This special edition of LIGHTWAVEPRO includes a sample of some of the articles found in every issue. We hope you enjoy it!

John Gross is a supervising animator for Arblin Imaging, a LightWave-based effects house in Hollywood, Calif. His work can be seen on such shows as Star Trek: Voyager and seeQuest DSV. E-mail him at jgross@netcom.com.

TABLE OF CONTENTS

- 4 Recycling Objects** by Grant Boucher
Learn about the proper uses of the subdivide and smooth tools.
- 6 Rules of Lighting** by John FK Parenteau
Learn the basic guidelines to successful lighting
- 8 MetaForm Magic** by Ken Stranahan
Building a car is easier with LW3.5's new modelling tool, Metaform
- 10 Your Friend the Null** by Glen David Miller
A new perspective on Null Objects and their uses.
- 12 Lens Flare Madness** by Mojo
With LW3.5 the lens flare has many features that makes it easier to use
- 14 Great Balls of Fire** Colin Cunningham
A few ways to create realistic fire and effects in LightWave.
- 16 You Bonehead** by Dan Ablan
Bring life to your inanimate objects with freeform deformation tools.
- 18 Mighty Morphin'** by James G Jones
Discover some of the many functions of 3D morphing.
- 20 Simple Space Stuff Part 1** by Mojo
How to put together a convincing LightWave scene in outer space.
- 22 Simple Space Stuff Part 2** by Mojo
Learn how the special effects of Babylon 5 come to life.
- 24 LightWave 101** by Taylor Kurosaki
Course 1A – a step-by-step lesson in LightWave object cleanup.
- 26 LightWave 101** by Taylor Kurosaki
Course 1B – Basic Modelling Strategies.
- 28 LightWave 101** by Taylor Kurosaki
Course 1C – effective surfaces for photorealistic environments.
- 30 LightWave 101** by Taylor Kurosaki
Course 1D – how to animate using spline controls.

LIGHTWAVEPRO

an Avid Media Group, Inc. newsletter

LIGHTWAVEPRO
An Avid Publications Newsletter

Editor
Managing Editor
Editorial Coordinator
Associate Editor
Art Director
Art/Production Coordinator
Production
Circulation Director
Circulation Assistants

John Gross
Jim Plant
Joan Burke
Corey Cohen
Helga Nahapetian Taylor
Kristin Fladager
Sergio "Berimbau" Miller
Sherry Thomas-Zon
Tracy-Ann Sparks
Debra Goldsworthy

Contributing Writers

Group Publisher

Editorial Offices: Avid Media Group, Inc.
273 N. Mathilda Avenue
Sunnyvale, CA 94086
Telephone (408) 774-6770
Fax (408) 774-6783

Dan Ablan
Joe Dox
William Frawley
James G Jones
Mojo
Michael D Kornet

About the cover: A compilation of LIGHTWAVEPRO cover shots from 1994 and 1995 issues.

May 1994 image – Greg Teegarden, © 1994

August 1994 image – Ken Stranahan, © 1994

September 1994 image – PTEN Consortium, Inc. © 1994

October 1994 image – FASA Corporation, © 1993-94

January 1995 image – Greg Teegarden, © 1994

February 1995 image – PTEN Consortium, Inc © 1994

Powerful Resources

AT POWERFULLY LOW RATES

Treat yourself or a friend to the magazines experts rely on.

VIDEO TOASTER USER and **LIGHTWAVEPRO** are your guides to the ultimate creative challenges of the Video Toaster[®], the Flyer[™] and LightWave 3D[®] — together at the lowest rates ever!



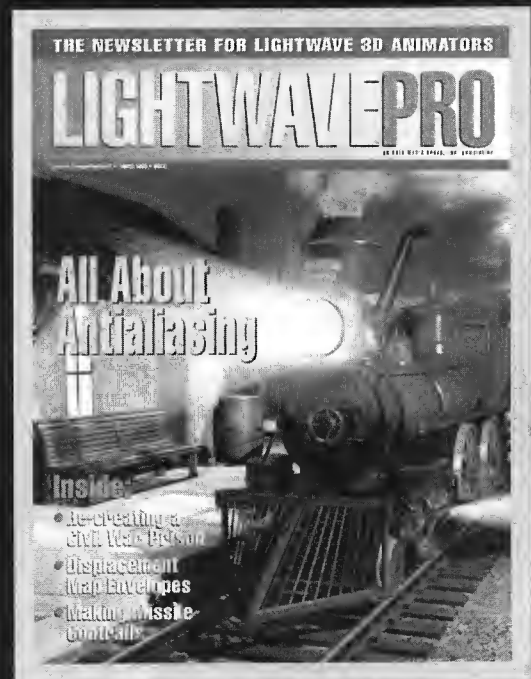
VIDEO TOASTER USER

It's the only magazine you'll need to create the most stunning video productions and LightWave 3D animation from the world's top video experts. Each issue is packed with in-depth tutorials, how-to articles, and distinctive and award-winning features. Get valuable tips, tricks and shortcuts to help you create your most creative video production to date.

One year (12 issues)

Special Rate US\$60

(\$24 + \$36 postage)



LIGHTWAVEPRO

Get the most out of LightWave 3D with the only newsletter dedicated to LightWave animation. **LIGHTWAVEPRO** promises a comprehensive blend of information, advice, professional tips and creative ideas to help you make the most of your next animation. And enhance your learning experience with supplemental disks full of ARexx scripts, objects, macros and more.

One year (12 issues)

Special Rate US\$84

(\$48 + \$36 postage)

For faster service, call and mention this ad

1.408.774.6770

or write to:

VIDEO TOASTER USER and **LIGHTWAVEPRO** • 273 North Mathilda Ave., Dept. 4 • Sunnyvale, CA 94086-4830 • fax 1.408.774.6783
Prepayment required on all overseas orders. Allow 6-8 weeks for delivery of first issue.

Recycling Objects

Proper Uses of Subdivide and Smooth

by Grant Boucher

If you're like me or most LightWave animators I've met, you've collected tens, if not hundreds, of megabytes of public-domain objects. These objects serve a good purpose by giving new LightWave users an opportunity to practice animation and surfacing (especially since modeling objects from scratch is a bit daunting to the 3D novice). Animators who move on to the professional level, however, often find that the objects found in public-domain libraries either lack detail or possess too much to be used efficiently in a production, given restrictions in memory and/or rendering time. Wouldn't it be nice if you could either increase or decrease the level of detail in an existing object, fixing imperfections and fine-tuning its rendering quality automatically? By using a few new Modeler tools, that is exactly what you can do. Dig into those directories with the date stamps from 1991—it's modeling time.

The Subdivide Tool

Hopefully, you've tinkered with the Subdivide tool found in the Polygon menu under Transform. If not, don't worry; we're going to start with the basics and work our way up.

First, we need an object. Load the apple into Layer 1 and notice the edges in wireframe (Figure 1). Let's make the apple more useful for closeup as well as medium and long shots. Phong shading smooths the polygons well, but since it doesn't actually add any detail to the object, the shape of the individual poly-

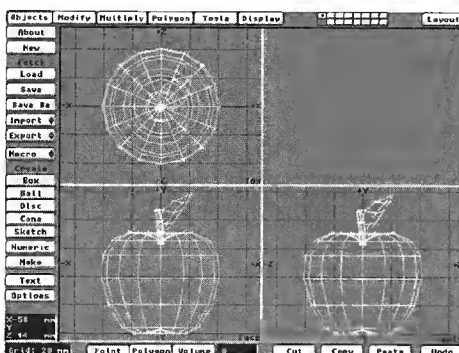


Figure 1: Load an object and its view edges.

gons is clearly visible. Before we proceed to subdivide the apple, we must first change all of its quadratic polygons to triangular polygons (the Modeler's Subdivide tool requires triangular polygons to work properly). Fortunately, the Triple tool is in the same Polygon/Transform section of Modeler as the Subdivide tool. Click on the Triple button or select Shift-t on the keyboard to triple all of the apple's polygons.

Once the apple is composed exclusively of triangular polygons, it's ready to subdivide. Select the Subdivide tool (Shift-d) and choose the Smooth option and OK. We'll talk about the Faceted option and the Max Smoothing Angle in a little bit.

Quite a difference, huh? The Subdivide Smooth tool actually creates a spline across the points of the polygon to be subdivided and finds the intersection with that curve for the creation of any new point. In Figure 2, the apple looks like it was initially created with more points. It also renders better in a closeup view. You might want to save this version as AppleSub or AppleMed, because it now has four times as many polygons as the original and thus takes longer to render. For situations that don't require this level of detail, the original apple object is far more efficient in terms of memory and time.

Maximizing the Smoothing Angle

Move to Layer 2 and load the LightBishop object from the Games/ChessPieces sub-directory. Again, this object is appealing when viewed from a distance. However, in a closeup or medium shot, it is clearly visible that the LightBishop's base has only 12 segments. While there are a number of more detailed chess-piece sets avail-

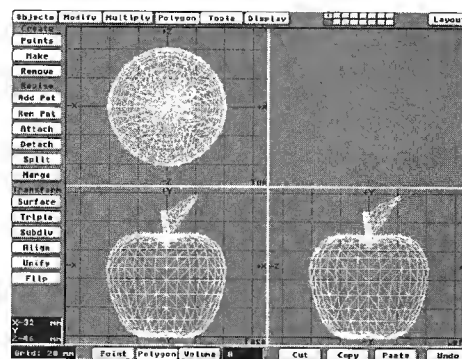


Figure 2: Notice the amount of points.

able in the public domain, let's invest a minute or two and see if this object can be improved.

Let's Triple and Subdivide Smooth the chess piece. Zoom in and notice that the little groove in the Bishop's head is filled with polygonal lumps (Figure 3), and the lower sections of the Bishop have buckled as if something was pushing it into the ground. That's not an improvement.

Have we discovered some limitation to the Subdivide Smooth tool? Yes and no. Remember I said that Modeler splined across points to make new points? The problem is there are places on the bishop where it shouldn't curve, such as the flat areas near the base and within the head groove. How do you distinguish between these areas?

The Max Smoothing Angle setting sets a threshold for the Subdivide Smooth

tool's spline. Hit Undo to get back to the tripled LightBishop and try the Subdivide Smooth tool with a Max Smoothing Angle of 44.5 degrees (just my first guess). Much better, right? With careful attention to the

**Wouldn't it be nice
if you could either
increase or
decrease the level
of detail in an
existing object; fix-
ing imperfections
automatically?**

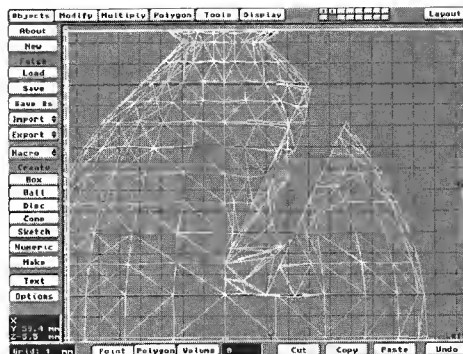


Figure 3: Note the lumps of polygons.

Max Smoothing Angle setting, even complicated objects with curved and flat regions can be subdivided safely. It helps to see the Max Smoothing Angle as the maximum angle that can be smoothed across two neighboring surface normals (extend the surface normals until they intersect with each other). The default value of 89.5 means any polygons whose surface normals intersect at an angle less than 90 degrees can be smoothed across. If there's a problem with an object that has multiple flat and curved areas, just select the curved areas to Subdivide and the flat, unselected areas can be ignored.

Warning: After I performed a Subdivide Smooth and viewed the object up close, I noticed some of the polygons were coming apart at the seams. I quickly hit the Undo button and performed a Merge Points/Automatic. My suspicions were confirmed as 84 duplicate points headed off to Modeler oblivion. When I Smooth Subdivided again, there were no problems.

Make sure your object is cleaned up before performing these manipulations. In this case, although two points overlapped, they each behaved differently when the Subdivide tool was applied. When the manipulation was finished, the points (and thus the polygons) no longer formed a perfect seal. Look for these things before and after you start reworking objects. It could save a lot of headaches during final renders, as seams like these rarely show up in low-resolution previews.

Preparing Objects for Bones and Displacement Mapping

One level of subdividing is usually sufficient to take any low-detail object to medium detail, which is generally acceptable for even professional animation projects. Objects already in medium detail usually don't

require any further subdivision unless they are to be further transformed with bones, object morphing or displacement mapping.

Try the following experiment. Load in the LightKnight object, Triple it and then Subdivide Smooth with a Max Smoothing Angle of 89.5. Take a close look at the flat, nose areas. You can see that the intermediate points were offset as previously mentioned, but in this case, I actually like the result. We get a rounder nose when we Subdivide Smooth again (this time using a Smoothing Angle of 44.5 to avoid mangling the flat, bottom polygons). Export this object to Layout as LightKnightHigh and set your Smoothing Angle (Surfaces panel) to 65.4 degrees (I like this smoothing effect; you might prefer something different), with lots of gloss and specular in your surface. The LightKnightHigh object has a clay-like, almost porcelain appearance and can now be animated with bones and produce pleasant results.

If you want to prepare an object with no curves or smooth surfaces, use the Subdivide Faceted tool. This is equivalent to using Subdivide Smooth with a Max Smoothing Angle of 0 degrees—meaning new points are created with no spline curves taken into account. The new points are created at the exact mathematical average of all the points around it. A good example is the EndTable object provided with LightWave. I suggest Subdividing with the Faceted setting at least three times before giving the object some bones.

Smooth—The Opposite of Jitter

Many animators have wrestled the mighty Phong when building or modifying objects. In fact, managing to position points and polygons to maximize a smooth appearance is one of the great tests of an expert modeler. Auto bodies and human faces are just some typical challenges faced by modelers every day, where the precise position of any single point can make or break the look of the entire surface. Prior to the release of Modeler 3.0, there was no weapon to battle this mathematical monster except hard work, a good feel for smoothing angles and a lot of render tests.

Fortunately, there is new assistance in the Points section of the Tools menu called Smooth. Its job is to look at all the points, checking to see if the polygons they are attached to can smooth across evenly when Smoothing is turned on as a surface attribute. What does that mean to you? Let's look at two examples.

I once worked on a series of animations for a client

where one of the principal characters was a little girl. The client wanted to make very subtle changes to her face, but I knew that every point I pulled might cause lumps and ridges to start rendering across the face. Since the client wanted to see the modifications rendered as we made them, I had to have some way of smoothing things out quickly. As we made changes to the face, stretching and pulling, widening and narrowing, whenever we were ready to do a render test, I ran the face through the Smooth tool. After a couple of short tests, we achieved the proper settings and were doing near real-time modifications to the character's face while the client watched over my shoulder (a nightmare for most animation jobs). I was able to pull the hair back and add new polygons to the sides of the head and face, and the Smooth tool would shift my new points to smooth evenly with the original points. For any animator who has to modify a lot of curved objects, the Smooth tool is a major timesaver.

The accompanying illustration (Figure 4) shows the power of the Smooth tool. After creating an asteroid using the same techniques described in the LightWave manual, combining Jitter and Subdivide/Smooth two or three times, I ran the asteroid through the Smooth tool with settings of Strength 2, Iterations 10. I find that low strength and higher iteration values seem to give me the right kind of smoothing for most modeling operations. The crumpled asteroid became a soft dough ball in about four seconds of work.



Figure 4: With smoothing, the asteroid become a dough ball.

Wrap Up

By using the Triple, Subdivide and Smooth tools, you can easily recycle your old library of objects. Combining these tools and perhaps adding Boolean features can give a lot of life to some of those public-domain objects that might be gathering digital dust somewhere in your archives.

LWP

Grant Boucher is an animator for Amblin Imaging. His work can be seen weekly on NBC's seaQuest DSV. In his spare time, Boucher is a science fiction and fantasy author and game designer. He can be reached on many national 3D electronic bulletin boards and through this newsletter.

Rules of Lighting

Benefiting From Contrast

by John F.K. Parenteau

The theory behind lighting is to produce an unnoticeable effect to compliment the given scene. Though that may sound like a dictionary definition, it is the ultimate goal of any cinematographer.

On set, the point of lighting is to illuminate the characters or environment in such a way that they appear to fit together as one cohesive unit. In CGI, this becomes even more crucial since a computer-generated model, by nature of being a model, tends to stick out.

In past articles, we have discussed the correct use of lighting and its appropriate positioning to give a natural effect to any scene. These basic rules of light placement (keylight, fill light, backlight and background light) will serve you as an animator many times. Beyond these simple examples, there is a whole realm of possibilities to consider when creating any new lighting design.

Following are a few rules:

Rule No. 1: There are no Rules

In real-world lighting, the limitations of the physical environment are the only rules to adhere to, and many of these are flexible as well. The sole goal is to avoid any obvious, unattractive lighting, unless it's called for. Lighting is creating the mood—whatever it takes to create the mood are your only requirements.

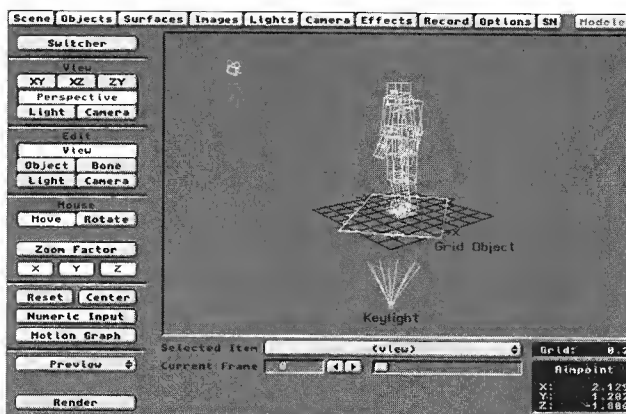
In a CGI environment, the “no-rules” rule can be carried to its extreme. Since light sources are invisible in front of camera, a computer animator can even place “fixtures” in the shot. If there is anything more frustrating in real-world lighting, it is avoiding a fixture appearing in the frame, a problem we never have to worry about.

Rule No. 2: CGI is Unforgiving

In the everyday world, a characteristic of lighting called radiosity works wonders. As light strikes a surface, that light bounces off and strikes another surface and another, until it decays to darkness. Quite often,



Figure 1: With no fill our character looks foreboding, but disappears quickly into the background.



Screen shot (1): A simple setup can yield dramatic results (see above).

placing a white piece of cardboard under an actor's face is enough to reflect enough light for an exposure.

In our CGI world, radiosity doesn't exist by nature. Any given situation where light would tend to bounce, reflect, diffuse or soften as it bounces off other surfaces must be created manually. Though raytracing a light can give the most realistic shadow, it also offers the harshest shadow. Techniques of softening edges must be employed manually to help reduce this hard edge.

These types of situations occur regularly in CGI and it is the job of the animator to adjust appropriately.

Rule No. 3: Use Contrast

Contrast plays an important role in all lighting. In the days of black and white photography, a skilled cinematographer manipulated contrast alone. As a research project, rent the movies “Nosferatu” and “Citizen Kane”. Both of these black and white films are excellent examples of how contrast can be just as effective in telling a story as the use of color.

A trick I used to play when shooting video was turning down the color on the monitor. If the shot looked good in black and white as well as color, I felt I had achieved an accomplished lighting setup.

Now rent the movies “Blade Runner” and “The Natural.” Though both of these pictures were photographed in color, the cinematographers (both masters, Jordan Cronenweth and Caleb Deschenel) use a great deal of contrast in color, as well as light and dark to tell their story.

A flat, front-lit subject not only looks bad in CGI, but just looks bad.

Rule No. 4: Help Tell the Story

Every image, scene or sequence tells a story. Even a still is conveying meaning or purpose. Try to design your lighting and camera motion to complement your story. Remember that the camera you are using to photograph the scene, whether it be real or CGI, is an actor as much as the character in front of the lens.

The cameraperson, is in full control of what the audience sees and doesn't see in the scene.

One of the most disconcerting things to see in demo reels we receive at Amblin Imaging is bad camera work. It is the most obvious detail to see, since even a bounding box preview will show errors.

If you are going for a mood, use the camera motion to help create it. The television show NYPD Blue is an excellent example of creating a mood for the audience. Considering the subject matter, the New York police force, the handheld feel they have established helps place the viewer in each scene, as if they were there.

In CGI, we have produced many a chase sequence where we used whip pans (panning drastically left or right to produce a blur in between) to help heighten the mood of the chase.

On the other hand, avoid gratuitous camera motion. If you are panning for no particular purpose, it probably isn't necessary. Keep in mind that, like the actors, the camera is garnishing a certain amount of attention as well. If you are whipping the camera around, banking and pitching for no reason, you are removing the audience from the scene.

Rule No. 5: If in Doubt, See Rule No. 1

Though rules two through four are valid and must be addressed, they are only suggestions of areas to take note. Any lighting design is dependent on the needs of the scene or shot. If you don't want contrast, don't have any contrast. Sometimes an evil character can be lit in a serene way to contradict the mood of the scene.

The preceding rules are only the tools of the trade and like any good tool, don't necessarily have to be used. Just be aware of them, and employ them when necessary. With these rules in mind, however, it is important to study ways to break up or diffuse light to create certain moods or effects. In the real world, a light source can be broken up in many ways. Various forms of diffusion, from high-tech "Tough Spun" to low-tech shower curtain, help to soften and mold



Figure 4: In a slightly more complex setting, we must take into account how light falls on our environment as well.

light before it reaches the character. In CGI, our methods are somewhat limited. By increasing or decreasing the cone size of a spotlight, we can increase or decrease the source and increase the soft edge of the light. Choosing between raytracing and shadow mapping assists in softening the edges of the shadow. There is little else we can do to directly influence the type of light created in LightWave.

But light is as much perception as it is reality, and careful consideration can create acceptable real-world effects.

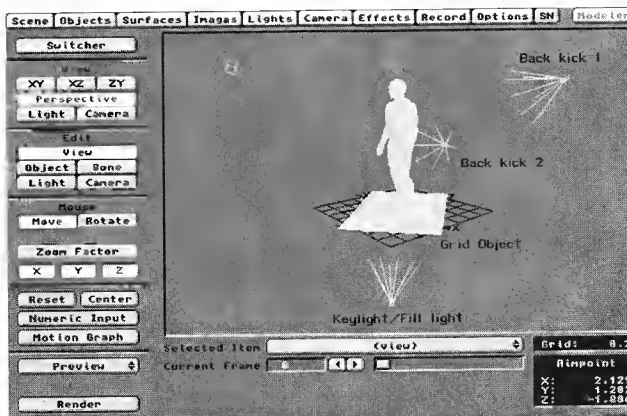


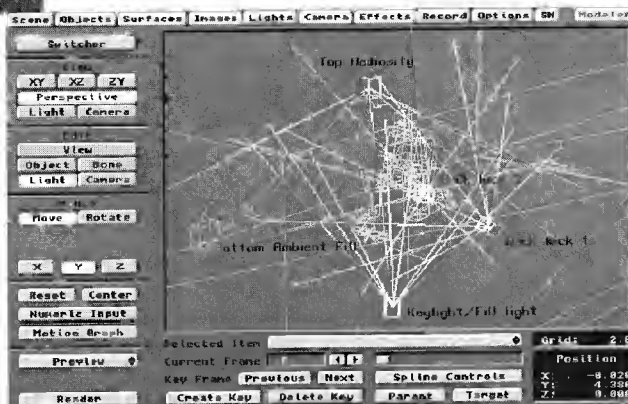
Figure 2: With the addition of a few lights, we should be able to separate our subject from the background.

Pass the Cookies, Please

In an effort to create an underlit mood, I modeled a grid in Modeler and placed it below my favorite subject, Humanoid from Crestline Software, and aimed a spotlight through it up onto Humanoid. Our grid, in real-world lighting, is known as a cukaloris. Any object used to break up light into a pattern is known as a cukaloris, or cuk (or cookie). It is important to note that the object creating the grid is a true 3D object. If a flat or single-sided object is used, errors may appear in your animation as the light strikes the surface at odd angles. If at all possible, extrude your cuk to avoid any rendering errors.

I've created a scene of the humanoid character against black Ambient Intensity (Lights panel) is set at 0 percent (as it should be). Placing our grid cuk at our subject's feet, I positioned a spot light slightly in front of our character and below the cuk, making sure the light cone was wide enough to cover the width of the grid. At 150 percent Intensity and a Falloff of 18 percent, a soft decay of light should occur, while still allowing illumination at the head. You can see the result in Figure 1.

Though the light falls dramatically up the body, the shadows are much too harsh to appear realistic. In addition, the shadows are completely black. Though this may seem appropriate, it isn't realistic.



Screen shot (4): Modification of the light positions to avoid unwanted light spill is a must in the actual environment. Note how the back lights have been lowered to avoid overfilling the floor.

In past lessons we have created a fill light at or near camera position. In an effort to create a more appealing effect, I cloned the keylight, removing any shadow options and reducing the Intensity to 20 percent. Though in most cases the Falloff should be set to the same relative amount of the keylight, so that the visible Falloff ring is in the same position for both lights, I have chosen to remove Falloff due to the low intensity of the light.

I have created a fill light effect that mimics light from the keylight spilling around the edges of the grid. If the fill light were placed



Figure 3: We have maintained the moody feel but no longer lose the shadows into the background.

beside our camera, its source would be invisible and questionable.

To correct the harsh shadow, I switched the keylight shadow option from Raytrace to Shadow Map, increasing the Shadow Map Size to 2048. Shadow mapping is a memory intensive process and your available RAM should be considered before using this setting. The larger your Shadow Map Size, the smoother the shadow edges created.

While these changes may be enough, I have taken our setup a step further (Figure 2). Considering our subject is standing on a grid, I placed a light behind and below Humanoid. This creates a back kicklight that helps pick our character out from the black background. In addition, another light was placed at shoulder height, behind and to the camera left of our man.

Though there is no motivation for this light, it assists the lower backlight in providing an added rim of highlight. Both of these lights were personal choices to add to the mood of the shot, and may or may not be necessary for your setup.

Now examine Figure 3 to see how the changes affected our character. Though it appears our subject is lit in an extreme manner, the added lights help the eye accept the drastic lighting without destroying the overall effect.

Into the "Real World"

It is one thing to produce a lighting example in a limbo environment. It is another to apply that same setup to an entire CGI setting.

Metaform Magic

Designing from Scratch

by Ken Stranahan

You're at your wit's end. There's one week remaining to finish the project and you still don't have the model built. In all probability, the spline modeling you have to do will be as easy as crocheting a car cover, and in the end it may not even fit the car.

Let's face it—every time you get a more powerful tool on your computer your life gets more complex. Splines are great because you can model anything, and it will only take three or four weeks. That is, if you're a good modeler. But come on now, I have a life and I like having my weekends free. Have no fear, there is a new tool called Metaform. For me, it has virtually replaced spline modeling, and the great thing is that it makes modeling work much easier.

With Metaform you can start with incredibly simple

objects. In fact, the first time I used Metaform I made my pre-metaform objects too complex. Also, it's not important to get the object's shape right the first time—you can adjust the shape in no time until it's perfect. This makes it a great tool for designing objects from scratch, and once you have the basic form worked out you can make dozens of variations in a matter of minutes.

So, let's discuss how to use Metaform. I think of it as being very similar to sculpting. You rough out a basic shape and then smooth it out. What Metaform does is subdivide an object and smooth out the polygons in between, just like sanding down the corners of a block of wood. The amount of smoothing is determined by the angle and distance between adjoining polygons. It sounds a lot more

complicated than it really is, so instead of discussing the details of this tool, let's just jump right in and build an object—a car. It's easier than you may think. This pictorial guide will take you through all of the steps.

1. Think of a car in its most simple form. It's a box with two wheel wells cut out and a box on top for the windows and roof. Even easier, make a profile of the car. To do this, enter Modeler and make a flat box (use the Numeric tool) with multiple segments for the wheel wells and roof. Enter 3 for the Y, 6 for the X and 0 for the Z. Figure 1 shows a sample segmented plane.
2. Now it's possible to select the polygons that are in the wheel wells and in front and behind the roof and cut them out. Take the polygons for the windows and

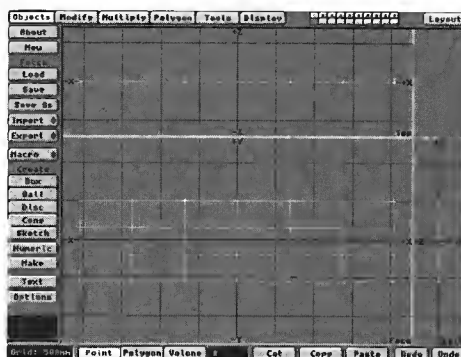


Figure 1: A simple segmented plane is where we begin.

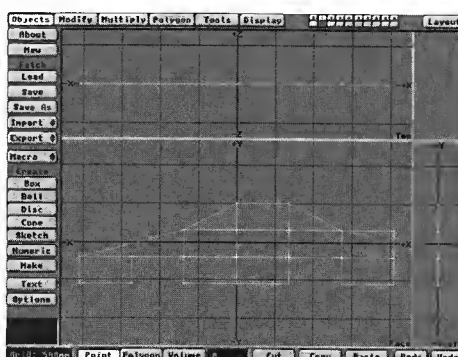


Figure 2: Points and polygons are deleted to form the basic profile shape of a car.

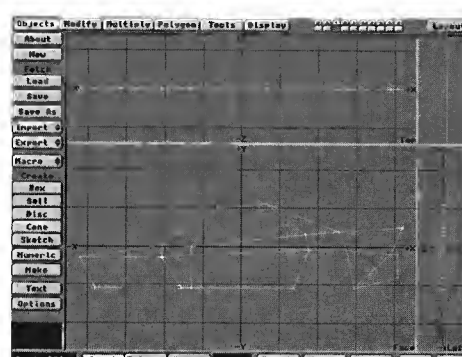


Figure 3: Our template after dragging some points and splitting a polygon.

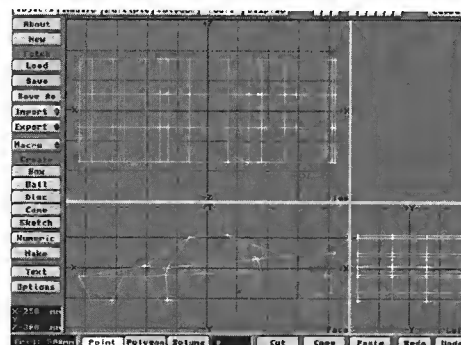


Figure 4: The extruded and centered primitive car shape.

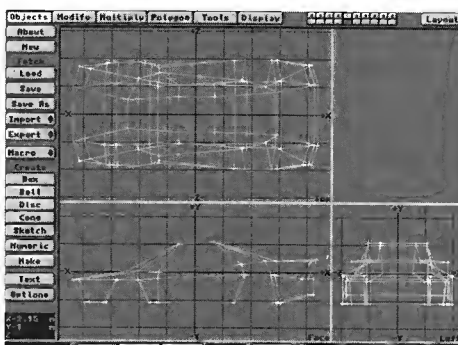


Figure 5: After moving, stretching and tapering our primitive car, we are starting to get a more recognizable shape.

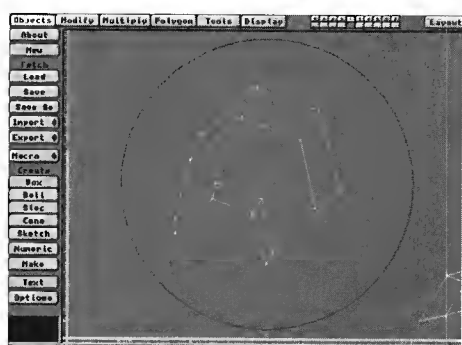


Figure 6: The static view of the wheel well section of the primitive car.

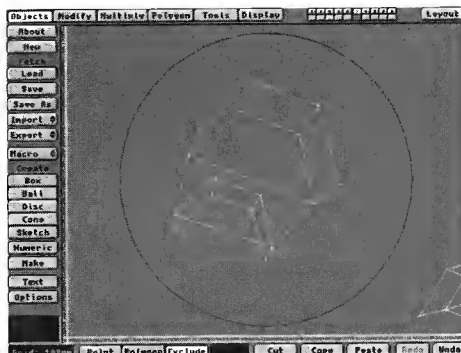


Figure 7: The inner points are selected on each side to create the inner wheel well side.

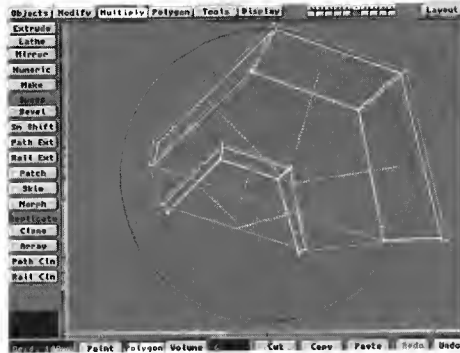


Figure 8: The sides of the wheel well are beveled slightly to create additional geometry.

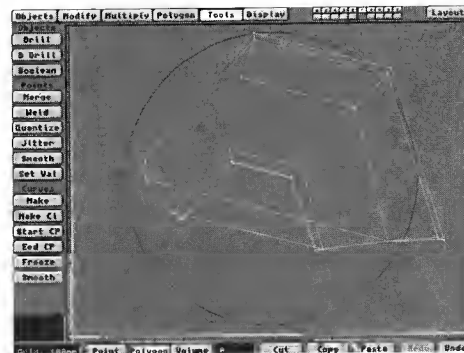


Figure 9: The sides after the points of the back polygons have been welded.

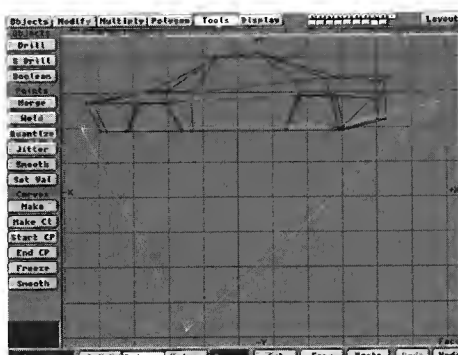


Figure 10: A solid box is created in order to stencil a row of polygons at the bottom of the car.

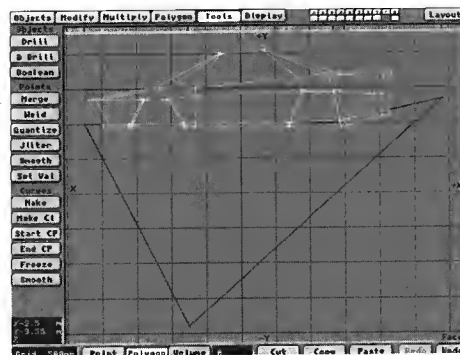


Figure 11: The newly added row of polygons of the bottom of the car.

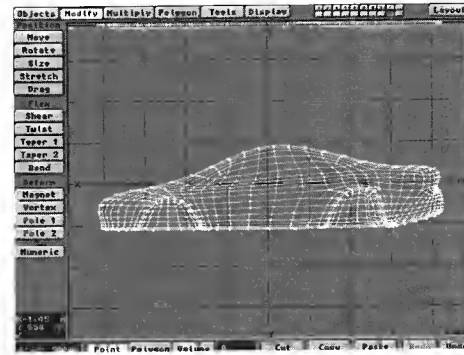


Figure 12: The final metaformed version of our car.

make them slant inward. Deleting the extra corner points so they become triangles should do the trick. Figure 2 shows the plane after sections are cut.

- Now that we have the basic sections, it's possible to adjust the points to make more of a car shape. The **Drag** tool (**Modify** menu) is perfect for moving the points around. It is a good idea at this time to split the polygons at the rear bumper and add a point to the bottom one in order to give more definition to the bumper. After adding the point, it is necessary to split the bottom polygon because Metaform will only work with three- or four-point polygons. Use the **Split** command in the **Polygon** menu to get your vehicle template to look like the one in Figure 3.
- Once the basic profile is finished it can be extruded to form the body. Do this with three segments so the body can have more shape (Figure 4). After extruding, the object can be shaped by stretching the points in the middle out toward the sides to give the sides some bevel. The roof and window points can also be stretched in to round them out, and the whole body can be tapered slightly toward the front. Then the row of points at the window line can be adjusted to give more curve to the windows (Figure 5).
- Finally, polygons can be made to fill the middle of the wheel wells out. Select the "wheel well" section, then select **Hide Uns** (**Display** menu) to hide the rest of the object.

Next, select the wheel well points that are "one section" in on the wells. Create a polygon inside each well and make sure that they face outward. Connect the

points on the bottom of the vehicle wheel well to create a floor. See Figures 6 and 7 for details. Choose **Unhide** (**Display** menu) in order to show your entire car again.

- By now the object is looking pretty good. OK, it actually looks like a box. But just a few more steps and it will look great.

If you try Metaforming it now (by pressing **Metaform** under the **Subdivide** menu, or **D**) you will notice that the wheel wells and bottom of the car will curve in. This is because when the car is metaformed, the closer the points, the tighter the curve, and the points are pretty spread out at this time.

- There are two tricks I use to take care of this. The first is to add polygons to the wheel wells so the edges stay flush with the body. For this, once again, select the wheel wells and single them out by using the **Hide Uns** tool.

First, select the three polygons that make up the sides of the wheel well that would surround the top, front and back of a tire. Then **Bevel** (**Multiply** menu or **b**) the polygons slightly (Figure 8).

Since it is really only necessary to have extra polygons on the outside edge of the vehicle, you can remove the polygons on the inside by welding the points (**Tools** menu) from the beveled polygons to the original polygons (Figure 9).

At this point, you will need to cut out unneeded polygons by using the **Polygon Statistics** requester (**w**) to select any two-point polygons and delete them.

- The second thing that can be done to add flatness to the bottom of the car is to stencil in a row of polygons

that are close to the bottom of the car. To do this, simply build an extruded box in a background layer that cuts across the bottom of the object and use the **Stencil** (**S Drill** — **Tools** menu) tool. Then remove any extra points that may occur at the corners of the box. This is a quick way to add extra sections to a simple object. See Figures 10 and 11 for more detail.

That's it! You can now Metaform your object.

If you have problems Metaforming the car, it may be because you have to clean up your object. Metaform doesn't like sloppy object building and you will get render errors if you aren't careful. So merge your points, look for two-point polygons and fix the holes in your objects before attempting any Metaforming.

Metaforming this vehicle will provide a basic car shape. From here you can stencil in windows, add wheels and surface the object.

Remember that once you have the basic pre-Metaform object, the points can be modified to create any kind of car. Adding an air scoop is as easy as beveling in a polygon on the side of the car. A rear wing, or spoiler, can be added by beveling two of the rear polygons up and connecting them across the top by making polygons across and then removing any unneeded polygons inside the wing.

Hopefully, this article has started you on the road to better Metaforming. I've included all of the layers of this Metaform example on next month's LWPRO disk for you to use as a starting point.

Ken Stranahan is an animator with Area 51.

Your Friend the Null

Fully Utilizing the Null Object

by Glen David Miller

You're probably wondering what an article on Null Objects is doing in the pages of LWPRO. In fact, though, it makes perfect sense. If I had to place a wager on the least used feature of LightWave, especially by beginners, I'd say it's the Null Object. Yet using nulls can help you avoid many near-disasters.

First of all, if you are keeping your LightWave program up to date (and you should be), from version 3.1 on, you will find an **Add Null Object** button above the current object pop-up menu.

At present, nulls save as part of the scene file, rather than as an object [Editor's note: To change the name of one of these nulls, select **Save Object**]. In 3.0 and all previous versions, a null object can be found in the objects directory as the only object not contained in a subdirectory, unless, of course, you've changed that fact yourself. In these versions, nulls are saved as objects.

What is a null object, anyway? In simple terms, a null object is an axis without the ability to render. It is the most basic object that LightWave understands. Through 3.1, the Null was displayed as a dot. In 3.5 (and 3.2 for you Raptor folks) the null took on the shape of a six-pointed axis marker.

So what good is a null?

Overcoming Gimbal Lock

First, there is axis, or gimbal lock. This problem is a side effect that occurs due to the way LightWave calculates rotations. If you were to position, for example, the spaceship object with the Pitch set to 90 degrees, you would notice that both the Heading and Bank move in the same direction. In essence, you have no heading control. Gimbals work the same way in the real world. If you are ever around one, give it a try. It seems impossible to overcome, and without a null object it would be. The normal way of getting around gimbal lock is to design your objects correctly in the first place. Spaceships and other vehicles should be built looking into the +Z axis. However, if you need to pitch too far, you can't avoid gimbal lock.



Figure 1

Here's how to beat it. After loading your object, take note of the pivot point. If you don't move your object, it should be located in the center of the screen. Then, using the method contained in your current software, load a null.

It should load in the same place as the pivot point of your object. The key is to parent the spaceship to the null, create a key at frame 0, then rotate the null 90 degrees in Pitch. Now you can reselect the spaceship and rotate around Heading Pitch or Bank. The gimbal is no longer locked.

The only other thing you need to remember is that when moving the spaceship, move it by means of the null. In other words, let the null control the movement of the spaceship, but let the spaceship itself control its rotations. Normally, you would want to keep the spaceship keyframed at 0 on X, Y and Z, but there may be times where offsetting the spaceship can produce desirable results. For instance, rotating a stationary null would produce a smooth, "orbit-like" motion for an offset spaceship.

When moving your objects throughout your scene, you may find it necessary to come up with a clever combination of rotations for your object and your parent null in order to give the movement the appropriate look.

Another Cool Moving Sheen

Moving sheens are often called for when performing logo work. Many sheens are achieved by moving an image map across a particular axis of an object. This is a good method, but what if you want a sheen to move around a logo? Again, the null makes this possible. Here's what you need to do.

After you have your logo (one with a bevel looks nice here), you will need to find a good image to reflect. An image with a lot of contrast, such as Figure 1, works best.

This image is loaded into LightWave (**Images** panel) and placed in the Reflection Image pop-up requester in the **Surfaces** panel. The rest of the surface of the logo should be set as follows:

Surface Color (Select your own, as the color will show through slightly)
 Diffuse Level 25%
 Specular Level 50%
 Glossiness Low
 Reflectivity 75%
 Soothing On
 Max Smoothing Angle 25°

All other values should remain at their defaults. Of course, depending upon your logo and your animation, you may need to change any of these values to obtain different looks.

Now, without moving the logo (and it should have been centered in Modeler), load a null object. Next, parent the logo to the null, then parent the camera to the null. You will now be able to move and rotate the camera to any position you wish. Remember to place a keyframe at frame 0 for both the logo and null object, as well as the camera.

For the rotating sheen to occur, we are going to have to rotate the logo, thus "moving" the reflection map around it, by rotating the null. Select the null object, click **Rotate**, and move your mouse around while holding down the left mouse button. It doesn't seem like much is happening except that your grid is moving around.

Look at the scene from one of the three orthographic views to see what is actually going on. Notice that even as the logo never seems to move (in relation to the camera), it is spinning around in space. Now, rotate the null so that the Heading and Bank have a setting of 360 degrees. Keep the pitch at 0 degrees. Create a keyframe for the null in this position at another frame. Let's use 150 for this example. After generating a wireframe preview, notice that the 150 frames loop cleanly. This gives you the ability to render only 150 frames, but you can output this sequence numerous times to film or tape for a longer animation.

If you only need a five-second, single-pass animation of your logo, you might place a Tension of 1 at keyframe 150 for the null object. This will slow the logo down to a stop at the end. After saving the scene, render it and you will see highlights moving around the surface of the logo. A positive setting for Heading and Bank will cause the sheen to move to the left and a negative setting will cause a move to the right. For this technique to work in its most basic form, the background needs to be set as a solid color, such as black. However, you are not restricted to using only a solid backdrop. You could build or paint a backdrop and load the image as a Background Image (Effects panel).

Null Offsets

A big area where null objects are helpful is when you need what I call "Offset Control." Offsets can, for one thing, make controlling multiple objects simple.

A great example of Offset Control exists in a computer-generated animation shown during this year's SIGGRAPH. In the animation, multiple rows of hammers move up and down in sync, banging their "heads" on metal anvils. Here is how to copy this technique using LightWave:

First, load the Hammer object from the Tools subdirectory. Notice that the pivot point of the hammer is toward the bottom end of the handle. That is a great position for the rotation that this hammer will take. Next, load a null object. [Editor's note: If you have version 3.5, you can easily add a null by selecting **Object** as the edit item and hitting the + key on the numeric keypad. Hitting the - key will ask to clear the selected item.] As in the spaceship example, it should load right on top of the hammer's pivot point if you have not

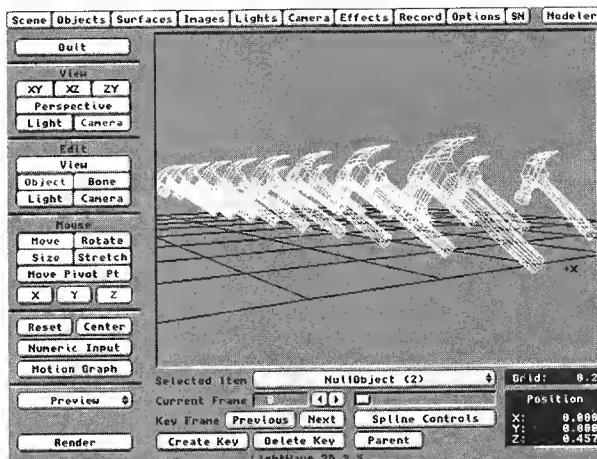


Figure 2

moved the hammer. Make the null the parent of the hammer object. Then create a key at frame 0 for both objects.

Set the hammer's heading for 180 degrees and rekey the hammer at frame 0. Next, enter the **Objects** menu and select the hammer as the current object. Click Clone Object and enter 8 into the field. After pressing return and exiting the Objects menu, you will see only one hammer. However, if you click on the Selected Item menu, you will see that eight hammers have been added. Select the second hammer. Then turn off Y and Z in the Move section and slide the second hammer along the X axis just to the left side of the original hammer. Create a keyframe for this hammer at frame 0 in this position.

Continue to move the next three hammers along the X axis to the left and create a keyframe at 0 for each. Then move the last four hammers along the X to the right and key those.

Once all of the hammers are separated, select the null and rotate the Pitch. Notice how the row of hammers moves in sync. There is also the added benefit of having to move only one object. Place the Null's pitch around -50 degrees and keyframe it at frame 0 and also at frame 50. Then pitch the Null down so that the heads of the hammers are flat against the "ground" (-90 degrees). Keyframe the null with this rotation at frame 20. With the null still selected, click on the **Motion Graph** button and select **Repeat** for the End Behavior. Click **Use Motion**.

Generate a wireframe preview to see the row of hammers move. Chances are that you will want to change your camera view a bit before generating the preview. At this point, save your scene as "RowOfHammers."

In the original SIGGRAPH animation, there were rows and rows of hammers. You, too, can have the rows and motions without a lot of effort. Make the Current Frame zero. Enter the **Objects** panel and select **Load from Scene**. Select the RowOfHammers scene in the requester and select OK. LightWave will ask you if you also want to load the lights from the scene. For this example, there is no need to do so.

In Layout, just like when the hammers were cloned, you should see only the original row of hammers. The new row is lined up with the original. Select NullObject (2) and, under Move, turn off both the X and Y axes. Move the second null back along the Z axis so that the second row of hammers is well behind the first row. Then keyframe the second null. Click the **Motion Graph** button (NullObject (2) selected) and, after entering the menu, select the Z Position as the Current Channel. Move the keyframe at 20 and 50 to the same Z Position value as frame 0.

Click Use Motion. Then make a wireframe preview. As you can see, there are two rows of hammers in sync (Figure 2). You can continue to add rows to your heart's content—or at least until you're out of RAM.

As you can see, using nulls can help you achieve many effects that otherwise would be difficult or impossible. There are a few more features I wish LightWave had that would benefit all the techniques discussed above. Some examples are the ability to cut, copy and paste sections of the motion graphs—to not only repeat the motion but repeat the motion a number of times and then stop, better still, to repeat a specific range of keyframes in the middle of a motion file a set number of times and then continue the file to the end. It would be really great if, to assist null offset animation, LightWave had the ability to morph one motion file to another. Even without these additions, however, the null can be a very helpful tool.

Look for all the scene files described in this article on next month's LWPRO disk.

LWPRO

The Complete Nebula Guide

continued from page 5

nebula. Also, because this object is self-luminous, you can turn off any lights to save some rendering time.

The Final Frontier

At this point, I call your attention to Figure 7. This is the technique Mojo described of mapping the nebula image onto a curved polygon object, which is sized marginally larger than the starfield. This then negates the need for a transparency map because every object lies in front of the nebula, even the stars. For this nebula object, I simply deleted a few top and bottom rows

and some of the sides of a globe-type ball, then flipped the remaining polygons inward.

The image in the color pages shows the result of using the "outside stars" nebula as before, but with the sky-filling technique, different regions can be magnified for different projects, including reusable nebulas.

To Summarize the Various Techniques:

1. PROCEDURAL/Interior Sphere/Fractal Noise Transparency or Luminosity

2. PROCEDURAL & IMAGE MAP/Interior Curved Polygon or Flat Plane/Fractal Noise Color/Nebula Image Transparency Map
3. IMAGE MAP/Flat Plane/Color Nebula Image Map/Nebula Image Transparency Map
4. IMAGE MAP/Interior Curved Polygon/Color Nebula Image Map

Engage!

LWPRO

Lens Flare Madness

by Mojo

For decades, if a director wanted cool lighting effects in a film, it took a team of rotoSCOPE artists to accomplish it. The glow of a spaceship engine? Easy. Simply create photographic blowups of every frame in the sequence, have the roto guy rough out in pencil the glow for each frame, then airbrush the finished product on acetate and photograph it against a light table. Of course, the finished film has to be sent to the optical department to be re-composited into the original footage.

These days, we just press the "lens flare" button. The history of LightWave's lens flare effect dates back to 1992, when Ron Thornton felt it would be an important feature in the production of the original *Babylon 5* pilot. Although it certainly helped contribute to the expensive look of the effects, the early lens flares required a lot of tedious envelope setting and manipulation. With the release of version 3.5, the lens flare now has a multitude of features that make it easier to use than ever.

Up and Down

Back in the early years, lens flares were simple, additive effects layered over the entire image after it was rendered. If we wanted a lens flare to go behind an object, it was necessary to carefully examine the scene frame by frame to determine when the flare should be obscured. An envelope then had to be created that ramped the flare up and down over the appropriate frames.

Now you can press the "fade behind objects" button in the lens flare menu and it's completed for you. When this is active, LightWave will figure out what objects are between the camera and the lens flare and ramp them automatically. However, circumstances may arise when this is undesirable: a lens flare should show through the base of a light-beam, or a transparent or gaseous object. In this case, by clicking off the "cast shadow" option, the lens flare will be instructed not to fade behind the desired object. If the object needs to cast shadows, you may need to resort to the old-fashioned method of hand-ramping a flare.

The biggest lens flare complaint has always been



Figure 1: The damaged Narn cruiser from *Babylon 5*. Perfectionists may want to create envelopes to make each flare shimmer a little to add realism.

the need to manually adjust the size of flares that move to and from the camera. Thanks to the new "fade-in distance" feature, this tedious enveloping is now a thing of the past.

To use this feature, set up a flare as you would normally. Give it a size you feel would be appropriate based on the effect and the flare's current distance from the camera and then type that distance into the flare's "nominal distance" box. Don't worry if you're having trouble figuring out the distance. Just type in any number and adjust either it or the flare size until the distance is correct. Once the flare looks fine, you won't have to worry about it again: LightWave will increase and decrease the size to maintain your settings whenever the flare (or the object it's parented to) is moved closer to or further from the camera.

The "fade-in fog" button has a similar effect, but obviously only works when fog is turned on. Once the flare has reached the maximum distance of the fog, the flare will be completely ramped down. I find that dissolving the flare out with an envelope can help or even replace this effect when necessary. Again, using the example of the engine exhaust, the flare would appear to shrink inside the engine object if fade-in fog was utilized. This would look wrong, while dissolving the flare out as the object dissolves into the fog would appear more natural. I would recommend fade with distance and a simple dissolve envelope on the flare instead of using fade-in fog at all.

Putting On the Squeeze

"Anamorphic squeeze" and "anamorphic streaks" mimic the look of photographing a bright light source with a wide-screen movie lens. When a wide-screen movie is made, the lens scrunches up all the information and the resulting image on the film is actually stretched vertically. The projector is fitted with a sort of "anti-lens" that stretches the image horizontally to correct it and fill a wide screen. Since lens flares are actually created by the lens itself and not in the scene, they end up being round on the finished anamorphic film. It is the expanding of the image when it is projected that gives this flare its ovalic shape. The blue streaks are simply an odd characteristic created by the lens.

Given this information, if realism is important to your animations, the anamorphic options should only be used in letterbox mode (or when compositing lens flares into existing anamorphically shot footage). Sure, few movie watchers know the whys and wheres about anamorphic lens flares, but these effects are never seen on anything other than big-screen movies (TV shows never have them since they are not shot with anamorphic lenses). In addition, they should only be used to simulate lens flares from bright light sources, such as flashlights or aircraft lights, and not effects (like engine exhausts). Of course, this only applies when realism is a goal—there really are no laws when all you want to do is make stuff look cool.

Tricks of the Trade

Headlights, engine exhausts, laser beams, explosions—these are some of the more obvious lens flare applications used by just about everyone who owns LightWave. However, after using it almost every day for the last several years, some of us old timers have come up with a few neat lens flare tricks that we feel can now be passed down to the young folks.

Ever want to rotate a lens flare? On a few occasions I have needed to make those neat ethereal streaks spin around in a heavenly fashion. The only way to accomplish this is to put a lens flare on a polygon and physically rotate it. You'll need to ren-

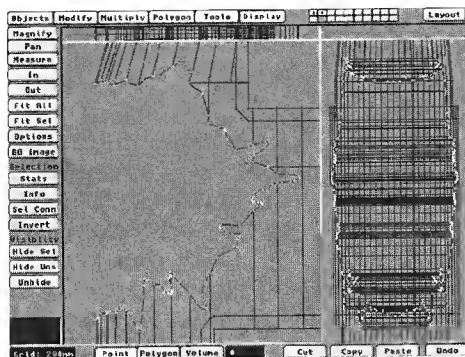


Figure 1A: Although not necessary, using point lights for lens flares helps you keep track of what's what, since point lights look like lens flares.

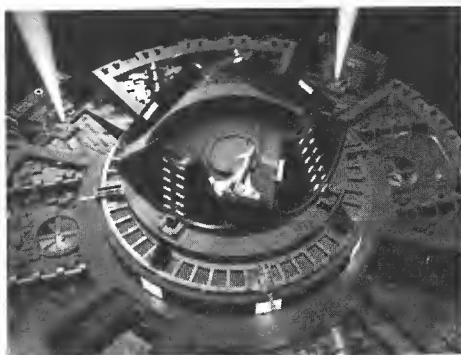


Figure 2: A spectacular image to be sure, yet somehow devoid of a certain something.



Figure 2A: Ah! Lens flares over the glowing panels do just the trick. What ponoche!

der a flare and save the RGB file to map onto a polygon. Make sure the flare is relatively small (maybe 60 percent) to keep the edges of the screen completely black, since you'll need to use this image as a transparency map as well. Unless the edges of the map are completely black, the edges of the polygon will be visible (use negative image for the transparency map). If the flare will get close to the camera or fill the screen, you will probably have to render the image as either high-resolution or print-resolution to avoid jagged edges in the flare streaks (for very pronounced streaks, make the light color zero and increase the size of the flare). This technique was used to create the Switcher's popular "flashbulb" effect, in which (you guessed it) a lens flare quickly spins into and out from the camera. This trick may also be desired to rotate the points of a star-filter lens flare.

When lens flares get big, it becomes much easier to see through them. This is a nuisance when trying to cover up a large area with a flare (such as an engine exhaust) and you always end up with an uncontrollably huge glow. How do you fix it? Double up your flares. Two flares in the same space can be controlled to create a strong center with a soft, manageable glow and not be too large.

One flare needs to be designated the hot spot (usually white) and the other serves as the outer glow (providing color). The hot spot should generally be about half the size of the outer glow and dissolved out as much as 50 percent. (The outer flare will add to the central one and bring up the apparent intensity—two 100 percent flares would create a flare with a 200 percent center and appear much too strong.) I usually turn off every flare effect except central glow—the deletion of especially the random streaks option creates a much softer, generally more pleasing effect. Also, remember that the incredibly

useful Lightswarm macro allows for the creation of double flares, so use it to save time when creating multiple sets of flares with similar attributes.

An incredibly cool, although time-consuming, lens flare effect is the heat fissure. By lining up dozens of low-intensity flares in close proximity, you can simulate a hot, glowing tear, welding streak, lava flow and many others. The idea is to put so many flares next to each other that they lose their circular shape and blend into one another, appearing to become a straight (or curved) line of light. Figure 1 shows a scene from Babylon 5 in which a damaged Narn cruiser has just been hit by an energy weapon. I wanted the metal to appear burning hot and knew the best way to do that was to line the damaged area with enough flares to simulate the effect. Since more than 100 flares were needed for the shot, the Lightswarm macro quickly became my best friend.

In Modeler, I isolated the points around the damaged polygon and copied them to another layer (Figure 1A). Using the drag points tool with the grid snap off, I positioned them to be more or less equidistant from one another. I then ran the Lightswarm macro and guessed at a lens flare setting. The key is to make the flares just large enough to cover the edge of the object.

The first rendering showed that the flare size was good, but individual circles were still apparent. This meant more flares. Back in Modeler, I added close to double the number of points I started with and ran another Lightswarm. This time the circles disappeared but the huge number of flares adding to each other created a center line that was too hot, so I had to re-run Lightswarm and give all the flares a 30 percent dissolve.

This procedure worked great. Then came the tough part—manually giving over 100 flares proper fade with distance parameters since Lightswarm

doesn't incorporate new features yet. Also important was making sure that only a few flares had light intensities. A nice orange haze over the surface of the ship was important for realism, but only a few evenly spaced lights with falloff were necessary. Absent-mindedly leaving effects flares with 100 percent light intensities is a common mistake, so keep an eye out for it.

The entire effect took perhaps half a day to get right, yet the results are clearly worth it. Always test out your effects with just a few flares to get a ballpark idea of how many you'll need and what their settings will be. This will save you a lot of rendering time and mean fewer visits to Lightswarm.

Believe it or not, lens flares can create subtle effects as well. My favorite is to place a dissolved flare (about 40 percent with outer glow only) over a visible lightsource or window. Figure 2 shows several luminous slits in a futuristic building's hatchway. However, Figure 2A displays the same slits with several lens flares placed over them, the result is much prettier and effective. The look is quite similar to photographing a scene with a fog filter and takes the harshness off a flat, luminous polygon. Use this to add just a touch of class to any scene.

I hope this gives some food for thought to all you flare-crazy animators. Lens flares look great, yet are so easy to use that few people give any thought to when they should be used. Like any other special effect, restraint is the key. After your 10th shot of a flare-encrusted UFO, the novelty begins to wear off. Use flares to add a little zest to your scene—not to hide poor modeling. If your shot doesn't look good without lens flares, you need to brush up on some other skills before you hit that oh-so-tempting button.

LWP

What's on the Disk?

This month's disk includes samples of Grant Boucher's ADProConvert.Rexx program, scene files from Glen Miller's "Your Friend the Null" and Ken Stranahan's "Metaform Magic" project file from last issue. Scene and object files from this month's issue and modeler patch to fix some bugs in Modeler 3.5 are also included. To order your LWPRO six-disk subscription, call toll-free 1-800-322-2843; rates are \$30 U.S., \$40 Canada/Mexico and \$50 overseas.

Great Balls of Fire

Hellfire & Brimstone, LightWave-style

by Colin Cunningham

Jerry Lee sure had the right idea, pounding away on his piano and then burning it to the ground for an encore. While that cat could torch a stage without batting an eyelid, mastering flames on the CG front is tricky enough to stop even The Killer dead in his tracks. There are a billion different ways to create realistic fire and heat effects in LightWave, so I'll primarily cover a few techniques used to burn up the screen on *RoboCop: The Series*. They may not be suited for all situations, but hopefully they'll inspire all you virtual arsonists out there to create your very own completely non-lethal pyrotechnics.

In episode 17, "Heartbreakers," our metal hero must battle corruption at the highest level and recover the stolen prototype for a weapon called the Heartbreaker, a microwave-emitting gun that can cause instantaneous heart attacks in its victims. The climax of the show has RoboCop being blasted by the device, an effect I was asked to develop. Since the beam of the gun is invisible (it's just a large microwave oven), I was halfway done already. I began work on the main effect: making Robo's body armor glow red-hot as he is bombarded by the deadly rays. After piling through a dozen harebrained schemes, including actually torching Robo actor Richard Eden (a suggestion not appreciated by the producers), I finally settled on using my old friend the lens flare. Though this was met by dirty looks and verbal abuse from my FX buddies, I thought it would work just fine. There is a saying I'm sure you've all heard at one time or another, "When in doubt, use lens flares," and for this particular effect, that adage couldn't be more true. Though impressive, lens flares tend to be overused, especially the kind that make your eyes water, with lens reflections and random streaks firing out of every corner. While lens flare abuse should certainly be punishable by death (or, more fitting to this article, being torched), I found flares to be a key element in simulating fire FX. When used creatively, they can add subtle realism to most scenes.

Glowing Armor

My first few attempts at making a nice glow effect involved positioning some flares over Robo's body

armor. Just as my co-workers suspected, the results looked like lens flares, each of them clearly visible. The flares also lacked the concentrated, uniform glow we desired; dissolving the glow 50 percent blended them together nicely, but the overall glow was too soft and not intense enough. The solution rested in the way the flares were spaced (Figure 1). Glow A consists of five flares arranged in a circular pattern; each flare is set at 100 percent glow with 0 percent dissolve. As you can see, each of the five flares is quite noticeable. Glow B, however, is made up of 13 flares arranged in a similar but tighter pattern. The flares range from 100 percent glow at the center to 40 percent glow and 20 percent dissolve at the outer edge of the circle. By adding more flares and placing them closer together, we allow the central glows to bleed together, forming a uniform and more intense glow. Using this cluster technique, it's possible to create complex shapes made entirely of lens flares.

There was more to the shot than I had first thought, however. Robo's armor had to gradually glow red-hot

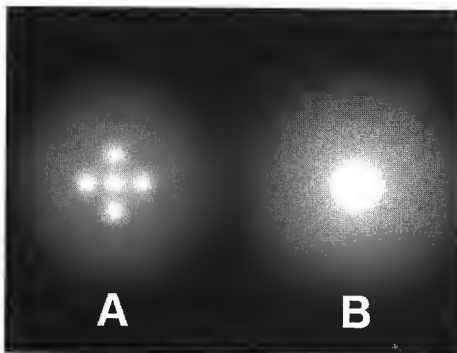


Figure 1: To avoid seeing individual flares (A), pack them tightly so their central glows "bleed" together (B).

as the camera tracked him stumbling backward into a wall. While it sounds like a difficult task, LightWave's new Background Layout Preview function simplified the process. I needed to lock the flare cluster to his chest as he stumbled around, so after generating a background preview from the footage of Robo, a null object was stuck to the center of his chest, traveling

with him every step of the way. I simply made keys for the null every three frames or so, fewer when necessary. Once that was complete, I parented seven lens flares to the null (five for the body, two for the helmet) and arranged them in a tight cluster pattern (Figure 3). To conclude, I set up dissolve envelopes for the flares to make them fade in gradually. Aside from a few minor adjustments every few frames, the flares moved with the null and the glow looked as if it was locked to Robo's chest as he struggled. As far as coloring goes, try setting the flare color to R 255 G 154 B 0 and turn on Central Glow and Red Outer Glow in the Lens Flare menu (Random Streaks and Outer Ring should be turned off). The combination of the two colors adds a realistic, fiery look to the flare and works nicely (see the color pages for the results).

Heat Ripples and Hotspots

Before moving on to creating actual flames in LightWave, there are a few more things we need to know about lens flares. The one problem I ran into when using flares was that the screen became too washed out and bright as more flares were added. Figure 2 shows two different approaches. Glow A is one lens flare set at 125 percent glow. Although the central hotspot is the correct size, it's not intense enough, and the haze around the glow completely washes out that half of the screen; it looks more like a

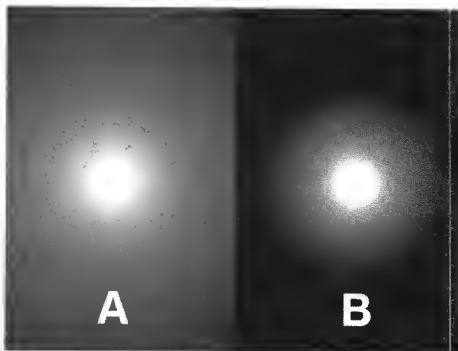


Figure 2: While single, large flares wash out the screen (A), many clustered flares produce a more intense glow without excess haze (B).

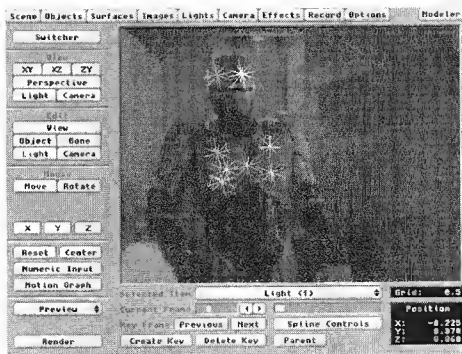


Figure 3: Parenting a lens flare cluster to a null object allows for more control when rotoscoping to background footage.

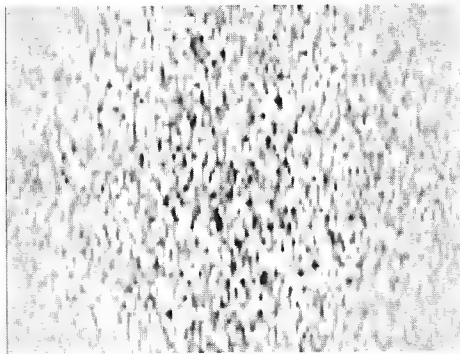


Figure 4: Use the default surface setting to better visualize your travelling bump map before using refraction.

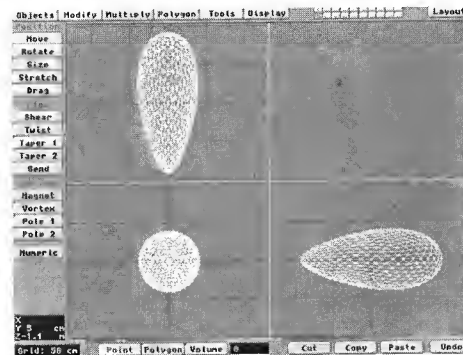


Figure 6: Taper the ball by a factor of 0.5 along the Z axis to get the general shape for the fireball.

light seen through fog. Glow B, however, is comprised of five tightly knit flares with 40 percent glow. This gives us an equally sized but more intense hotspot without the unnecessary haze that accompanies large lens flares. The desired effect is up to you, but hopefully you now have a better idea of how to manipulate lens flares and coax them into doing your bidding.

Although the flare method looked just fine, there still seemed to be something missing from the shot: Robo's armor didn't really look, well, hot enough. I thought nothing more of it and went to see *True Lies* with the rest of the department. We went for a few drinks after the show and talked about a few scenes that caught our fancy, particularly the ones with the Harrier jets. The heat signature they emitted really gave the impression that hot exhaust was blasting out of the jets. It was this element that could give my scene more realism, with this in mind, I awoke the next morning and got right to work adding heat ripples to RoboCop's glowing body.

The first step was Modeler, where I built a box 2.692 meters wide by two meters high by 0 meters deep (see last month's article "Interactive Refraction" for more details). Name the surface of the box "Screen" by hitting "q" for Change Surface (Polygon menu) and save this object as "screen." Hit "q" again and rename the box surface "Screen REF." Save this copy of the object as "Screen2" before exiting to layout.

Position the camera at 0,0,0 and make a keyframe for it by hitting return twice. Next, load in the "Screen" object and position it at X 0, Y 0, Z 3.2 before making a keyframe. The next step is simple enough: for my scene I loaded the background footage of Robo as an image sequence and planar-mapped it onto the screen along the Z axis (don't forget to hit automatic sizing). Set the luminosity to 100 percent and the diffuse value to 0 percent. Now comes the interesting bit. Load in the "Screen2" object and position it away from the first object at X 0, Y 0, Z 3.1 (remember to make a keyframe). Enter the Surfaces menu and set the "Screen.ref" surface as follows:

Color: Doesn't Matter
Specularity: 0%.
Transparency 100%
Refractive Index 1.25

Go into the BUMP MAP menu and set the values as follows:

Size X 0.021 Y 0.045 Z 0
Texture Falloff X 50 Y 30 Z 0
Texture Velocity X 0 Y 0.035 Z 0
Amplitude 325% Frequencies 1

Exit into the Camera menu and select Trace Refraction before rendering. Simply put, as the bumps move up the invisible refraction screen, they distort the image sequence on the screen behind just as heat ripples distort things behind them (Figure 4). A texture falloff was added because RoboCop happened to be standing center-frame and I wanted the heat signature to fade off gradually away from him. Because we had a Screamer at our disposal, I wasn't too concerned about rendering time (it turned out to be a couple of minutes per frame), but you may want to try faking refraction as mentioned in Dan Ablan's "Faking Refraction" (September 1994). I received better results using refraction, but not everyone can afford to wait an hour for a render.

Fireballs-a-Plenty

Now comes the part where we actually destroy things. At this point, Robo's armor is glowing more intensely by the second and it looks as if he'll be nothing more than a hunk of burnin' scrap if something isn't done fast. Just when things appear bleak, Robo reverses the polarity of the Heartbreaker beam, blasting a huge fireball across the room that incinerates

everything in its path, including the bad guy. Some real pyrotechnics were actually set off during filming, but the fireball itself was to be a LightWave effect. This was by far the easiest and most enjoyable part of the scene.

While in Modeler, select the Ball tool in the Objects menu and create a Level 3 Tessellation with the default radius (hit "n" to enter numeric values for tools). Once created, use the Stretch tool (Modify menu) to stretch the ball by a factor of 1.84 along the Z axis (Figure 5), again using "n" to enter numeric values. Next, use the Taper 1 tool (Modify menu) to taper the object by a factor of 0.5 along the Z axis, make sure the sense is set at "-". We've just created the general shape for our fireball (Figure 6). Hit "q" and name the surface "Fireball-OUT." Now select all the polygons in the object by holding down the right mouse button and lassoing the entire object, you can make sure all polygons are selected by hitting "" for Select All Connected. Copy and Paste the object in the same layer and while the polygons are still selected, Size the object (Modify menu) by a factor of 0.95. Keep the polygons highlighted and hit "q" to rename the smaller object "Fireball-IN." Although we now have two fireballs (one inside the other), save the layer as one object called "Fireball" before exiting to Layout.

Although the shape is definitely important, it's the surfacing of an object like this that can determine its success. That said, load the fireball object and enter the surfaces menu. I achieved a nice effect by surfacing as follows:

Fireball-OUT

COLOR 226 225 155
ADDITIVE On
LUMINOSITY 0% with GRID

Texture as follows:

FALLOFF X 0, Y 0, Z 85
VALUE 90%
LINE THICKNESS 2
DIFFUSE 0% with GRID

Texture as follows:

FALLOFF X 0, Y 0, Z 90
VALUE 100%
LINE THICKNESS 2
SPECULARITY 0%

TRANSPARENCY 4% with FRACTAL NOISE

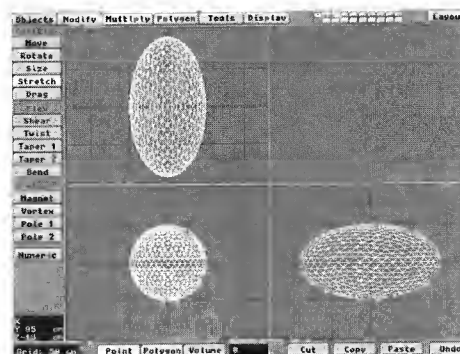


Figure 5: Stretch the ball by a factor of 1.84 along the Z axis

You Bonehead!

Part I: A Beginner's Look at Bones

by Dan Ablan

Ahh, yes. Bones. By now, most of us know what they can do. We know you can make inanimate objects come to life. We know that Bones can make solid objects bend and deform. But not many people use Bones on a regular basis, because they're either too tedious to set up, or when you do, weird things happen to your object. Well, believe it or not, they are not nearly as complicated as you may think.

If you've worked on any other 3D platform, you'll see that only a few, like LightWave, have Bones. Bones are an advanced form of free-form deformation that enable the animator to create fluid character animation. You can even make smooth-flowing curtains with just a few Bones. Bones will also allow you to simply deform an object, and in a sense, model in LightWave's Layout. It's good to think of LightWave's Bones as handles. You probably already know all of this, so let's move on.

I thought I'd put together a tutorial from a fun animation I did when I first got LightWave 3.0 and Bones. Arnie Boedecker and I have a series of Bone tutorials we're putting together for LWPRO, based on a big project just completed. There were quite a few tricks we picked up when doing those animations (like those flowing curtains I just mentioned), so keep your eyes on future issues of LWPRO.

The scene I'm talking about in this article is probably one of the most common animations people have tried using Bones, thanks to, you guessed it, the Listerine ads. Those sensational animations were produced by a company called PIXAR, but there is no reason you can't put together an animation like that at home using LightWave and Bones. Just so no one is misled, we'll call our product Blisterine.

Proper Construction

- The first thing to do is build your bottle. This type of object is pretty easy to build using **Lathe**, so I won't go into details. The way I built mine was by going to the store, buying a bottle of Listerine, and placing it on top of my computer. Then I built it. A more mathematical approach can be used by measuring. Plus, you can grab a frame of the bottle, then trace over the image in Modeler.

- For the label, I used Modeler's **S Drill, Stencil** feature (**Tools** menu) to stencil in the label area on the bottle. To create the label image, I found the ToasterFont that looked as much like the original label's fonts as possible. Then I made sure that the spacing was the same, and the size of each section of words. What the words read, however, is not at all what's on my bottle. I composed black letters on white in CG, then imported that frame to ToasterPaint and drew the border. I saved this as a full-size image, so that in the animation, when the bottle jumps forward and lands in front of the camera, you can clearly read the label. It takes up more RAM in LightWave, but it's well worth it when everything is sharp and clear in the final piece. For this tutorial, you can use the pop can supplied in the Toaster's objects directory to save time.

With Bones, anything you want to deform needs to be made up of many polygons. For instance, if you took a storm window off of your back door and tried to bend it, it wouldn't bend at all, or it would crack. But, if you took a screen off of that door and tried to bend it, it would bend very smoothly and easily, right? That's because the screen is made up of many segments. Think of your objects the same way.

In Modeler, to create many segments, use **Subdiv** (**Polygon** menu). You first need to **Triple** (**Polygon** panel) anything before you subdivide. [Editor's note: With Modeler version 3.5, you can subdivide three-sided or four-sided polygons.]

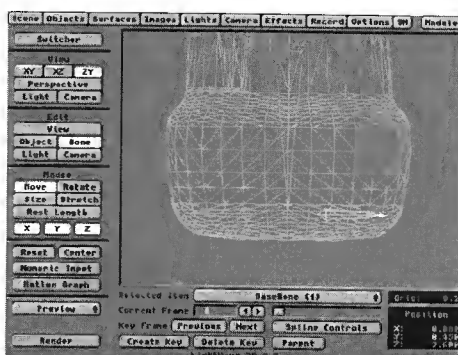


Figure 1: The first base Bone in place as an anchor.

You should also make sure to check for non-planar polygons (**Polygon Stats-Display** menu) and triple any that you find. If your whole object is composed of triangles, you will have no non-planars. This is best for objects that will be deformed in Layout.

- Once this is done, **Export** your bottle to Layout, saving it as bottle.sbdv or soemthing similar. Remember that Bones affect the points of an object, not the polygons, so when you start moving your Bones, don't be confused by the bounding box of your object staying in place.

The Setup

One thing to remember is that Bones are saved with the scene file, not the object. Bones have an unlimited influence on an object. If you don't limit the influence with the **Limited Range** setting in the **Object Skeleton** sub-panel, your entire object will be influenced. The more common way of taking care of the influence is by placing other Bones around the object.

On my Blisterine bottle, I've placed four Bones in the base, renaming them base Bones. Those base Bones don't move because they will act as anchors for the object. Because I'm going to move and rotate the other Bones, the anchors hold the bottle on the ground. The result is a bottle that bends and twists. If those base Bones weren't anchoring the bottle, the entire object would move.

Under the **Objects** panel in Layout, you'll see another button labeled **Object Skeleton**. This is where Bones are loaded, cleared and renamed for your scene. In order to place Bones to a specific object, that object must be selected before you go to the skeleton panel.

- Click **Add Bone**. Rename that Bone "basebone." Return to Layout. You'll see what looks like a squared necktie. That's your bone (Figure 1). This Bone has no effect on your object yet. Select **Continue** and return to the Layout window.
- The next thing to do is choose the object, select **Bone** and choose **Rest Length** from the mouse function control area in Layout. Do not confuse Rest Length with Size. The size will actually change the shape of the object, whereas the rest length determines the amount of influence the Bone has.

The larger the Bone rest length, the more influence it will have on your object. Since I'm placing this first Bone as an anchor, it's sized to half the length of the base of my object. When I add the next three Bones, they will be the same size as well. From this point, you can repeat the process, or if you are working with LightWave 3.5, the **Add Child Bone** feature in the skeleton panel can be used.

This function will clone the Bone and parent it to the currently selected Bone, so you don't have to reset the rest length. **Add Child Bone** is really helpful when putting bones in a snake- or rope-type object.

- Now, rotate the Bones you just added so that you have a base Bone in four places on the base of the bottle—north, south, east and west—and create a keyframe at zero for each. So far, these Bones still have no effect on your object. To make them have influence, press (r) to determine the rest position and direction. If you now move or rotate these Bones, they will affect the object. However, we don't want to move these. They're the anchors, remember?
- Go back into the Object Skeleton panel by pushing (p) on the keyboard while a Bone is selected. Add another Bone (not a child Bone). Rename it "bottombone." This Bone will be used to control the bottle's lower half. Change the Rest Length and rotate it 90 degrees vertically. Think of this Bone as a leg for our bottle.



Figure 2: The neck Bone in place.

- Add another Bone and rename it "chestbone." Place this one in the top middle portion of the bottle. Finally, add only one more Bone, renaming it "neckbone," and change the rest length, then rotate it to fit it in the neck area (Figure 2). Once your Bones are in place, type (r) on the keyboard and create a keyframe for each bone.

If for some reason you set a Bone wrong, you can deactivate the Bone from the skeleton panel by de-selecting **Bone Active**.

- Now, save the scene. Get into the habit of saving the initial Bone setup. That way, if your bone movements get totally out of hand, you can reload the setup scene, and not have to deal with resetting all of those Bones.

Simple Surfacing

I've tried surfacing the Blisterine bottle a number of ways, and recently came across a way to save time by not using traced reflections. You'll definitely need to trace refraction for a glass bottle, though. By loading the Fractal Reflections image and using it as a **Reflection Image** with 10 or 15 percent **Reflectivity**, you'll achieve a nice abstract reflected look, and help your object appear more real. It's a great setting for getting clear plastic to look convincing.

Originally, I made both the bottle and liquid inside it. Each had different surface properties and different refraction indexes. To get true refraction, the light would need to go into the bottle at one setting, pass through another, and leave at yet another setting. Polygon count was high, so just for grins and giggles, I tried something else. I resurfaced the body of the bottle with a different name. I took the liquid out of the bottle and kept one refraction level for the whole thing. Then, I surfaced the body section the color of my liquid, with the neck portion of the bottle fully transparent (see below). In the final animation, the effect was barely noticeable, and saved time rendering because there were fewer polygons and less refraction to calculate. The downfall of this procedure is that the liquid in the neck area won't slosh around.

Bottle Surface Settings

Color	184, 122, 46
Diffusion	90%
Specularity	80
Glossiness	High
Reflectivity	10%
Reflected Image	Fractal Reflections
Transparency	60%
Color Filter	On
Edges	Normal
Smoothing	On
Refractive Index	1.3

Neck Surface Settings

Color	200, 200, 200
Diffusion	80%
Specularity	80%
Glossiness	High
Transparency	100%

The Motions

Now it's time to animate. One thing to learn is to be patient with your movements. Too often, animations are rushed to get a certain amount of movement accomplished within a certain time frame. Work to avoid that. With Bones, timing is everything. The best way to know how to move your character around is to study motions from everyday life. When creating a more cartoonlike character, the movements are exaggerated, so watch how traditional cell animations are drawn and animated.

From here, it's totally up to you how everything will work. For my bottle, I decided to make it look around, then lean back, lurch forward and flip in the air, land-

ing in front of the camera. This became a little tricky, because the Bones bend the object, but keyframes needed to be set for the bottle as well. The chest Bone moves back, making the bottle lean, then it quickly swings forward, and at that point, the bottle starts to flip up in the air. Keyframes for the bottle needs to be set when it starts its jump, in mid-air, when it's upside down, and again when it lands. To add to the whole motion, consider stretching your object on the Y axis, when it lands, while moving the Bone forward quickly, then back, as if the bottle almost loses its balance when landing. The motions for this movement are on this issue's disk with a pop can object ready to animate.

What really helps sell the character, whether it's a bottle, can, glass, or even a square box, is proper keyframing of the motions. To take your animation one step further, consider adding hands, feet or weapons to your object. Since I'm from Chicago, I thought it appropriate that the Blisterine bottle was connected—you know, with the Mob.

Once you've set up the object's movements with Bones, save the scene. Then create the surrounding scene, and simply use the **Load From Scene** (objects panel) feature to import your Bone sequence. I've put my mobster in a scene, but made it black and white to fit the time period. By parenting a machine gun, among other objects, and keyframing that, the illusion is created that the bottle is holding the gun (see color pages).

Remember This

Bones are a very powerful tool. What I've learned in working with them over the past year and a half is patience. The latest project with bones went so well because the entire animation was storyboarded first. We knew what we wanted to accomplish, and when it came time to animate, everything fell into place.

By the way, there is a Ninja and Rambo Blisterine bottle animation in the works.

LWP

Dan Ablan is a LightWave animator for AGA in Chicago. He's done work for The Dial Corporation, Kraft Foods, NBC in South Bend, and others. He can be reached at (312) 239-7957 or via Internet at dma@mcs.com.

Beginner's Steps

- Think of Bones as handles to pull and push an object.
- Objects need to be tripled, and sometimes subdivided, to bend properly.
- Rest Length sets the amount of influence of the Bone.
- Sizing a Bone changes the size of the object, not the Bone.
- Use Add Child Bone (LightWave 3.5) to save time.
- Always save the scene once Bones are in place.
- Save scene with Bone movements under a different name.
- Be patient with the amount of movements.
- Stretch the object while moving Bones for added characteristics.

Mighty Morphin' Morphing Tricks

by James G. Jones

What is morphing? Well, this article doesn't have anything to do with 2D image morphing, the type made popular (perhaps too popular) by Michael Jackson's music video "Black and White." And, although morphing is "mighty" powerful, I'm certainly not talking about those annoying adolescents in multi-hued spandex who gallivant around the tube on Saturday morning.

This is about 3D morphing: the ability of LightWave to change the geometry of one object to match the shape of another.

Limitations

There are a few limiting factors. First and foremost is the requirement that the source object and the target object have the same number of points. Trying to morph an eight-point box to a 34,000-point Tyrannosaurus will not work.

Furthermore, with few exceptions, the target object should be modeled directly from the source object. In practical terms, this means you create a source object in Modeler, then bend, stretch, twist, move, resize, rotate, push, pull, dent, drag, kick, mangle and otherwise reshape your original object into its morph target. In some instances, you might have to resort to moving points around individually.

One very important thing to remember is to save the source object before getting out the hammer.

Another limitation is that as the points of the source object move to their new positions in the target object, they move in straight lines. However, there is a way around this that we'll get to in just a moment.

Digression on Theory

Many of us have a difficult time understanding just what is going on with source objects, target objects, metamorph levels and envelopes. If I had to describe the basic principle involved in a succinct manner (and I guess I do), I'd say that you should think of a morph target merely as data that the source object "looks" at to determine what shape to become. If you keep that thought in mind, I think everything else will be a bit more understandable.

How to Set Up a Morph

It's all about percentages, envelopes and who is targeting whom.

Say you have three objects: a bust of Bill Clinton, a mailbox and a back half of a horse. They were all modeled from the same object and have the same number of points.

This is obviously a hypothetical situation, so just pretend you're doing the following steps:

- Load the three objects into Layout. Set the mailbox and horse's butt to 100% **Object Dissolve** (Objects panel). You could just move them out of camera view, but I like to use Dissolve because it's right there in the objects panel.
- Set the **Metamorph Target** for Clinton to be the mailbox, and set the **Metamorph Level** to 100%.
- Set the **Metamorph Target** for the mailbox to be the horse's hiney, and set the **Metamorph Level** to 100%.

Now close the Objects panel. What do you see? Yes, that's right: Bill Clinton is a horse's ass. (I've always wanted to say that.)

But why? OK, here's what's happening: the mailbox is 100% morphed into the shape of the horse's behind. The bust of Clinton is 100% morphed into the shape of the mailbox. But since the mailbox is now shaped like the equine posterior, that is the shape to which William is morphed. Get it? If not, read the preceding paragraph again. Slowly.

This is called a "chain" of morph targets. In other words, object 1 is morphed to object 2. Object 2 is morphed to object 3. Object 3 is morphed to object 4. (Object *n* is morphed to object *n*+1, for you math heads.) This can go on for quite a spell—up to 16 times for any one source object, according to the manual. Rumor has it, though, that you can actually have far more than 16. I, for one, am not about to set up a 97-object chain of morphs just to find out the real maximum. If you have excess time on your hands, feel free. [Editor's note: For those of you without excess time, the limit is 40!]

To control the morphs over time, you use envelopes. For example, you'd set up an envelope for

the Clinton bust object with a keyframe at frame 0 equal to 0%, and a keyframe at frame 15 equal to 100%.

Then you'd create an envelope for the mailbox that goes from 0% at frame 0 to 0% at frame 15 to 100% at frame 30.

The resulting animation would show Clinton's head at frame 0, changing to the mailbox at frame 15, changing to the horse's hindquarters at frame 30.

Morphing Real Objects

Enough theory...here's a slightly more complex (and less political) example that you might actually find useful.

How about morphing a flat plane to a sphere?

Having an animation where a flat plane changes to a sphere is one of those things that sounds easy to do at first, but turns out to be a bit tricky in reality. Here are a couple of approaches I've worked out: one solves a problem with the Wrap-To-Sphere macro in Modeler, is very simple and gives OK results. The other is a bit more complex but looks much better.

Method One

- In Modeler, use **Box** (Objects menu) to make a flat plane that is twice as wide as it is tall. For this example, 2 meters wide (along the X axis), 1 meter tall (along the Y axis), with no depth (along the Z axis). Be exact. Use **Numeric** (n) to enter the pre-

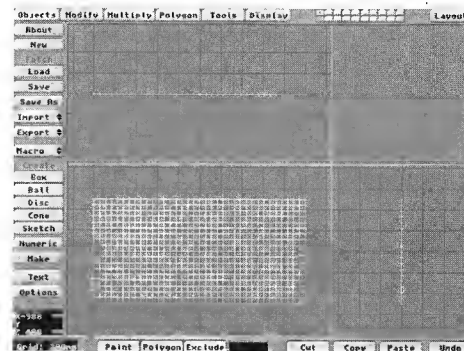


Figure 1: Make a flat plane in Modeler, 36x18 subdivisions.

cise values. Enter 36 subdivisions along the X axis and 18 subdivisions along the Y axis. You should end up with a flat plane that looks just like Figure 1. Use the Center macro to center the plane if you built it off-center.

- You will notice that the polygon's surface normals will be facing toward you (toward the negative Z axis). **Flip** them all (**Polygon** menu or f) so they are facing away from you (toward the positive Z axis). This will make sense in a moment, trust me.
- Save this object as "FlatPlane."
- Now go to the **Macro** button and choose "Wrap-To-Sphere." When the requester appears, click on the **All** button and set the inner radius to 0.5 meters. Click on **OK**.

You will now have a nice-looking sphere, and you will notice that the polygons are facing the way they should—outward.

- Save this object as "Sphere."
- Now load these two objects into Layout and set up a morph from the FlatPlane to the Sphere over a period of 30 frames. Don't forget to set the Target object (Sphere) to 100% dissolve. Make a wireframe preview and the first problem you'll notice is that the flat plane turns inside-out on its way to the sphere shape.

This is not good. However, you could always run it backward and do an animation of an imploding grapefruit.

By the way, I have no idea why this macro flips the points around like this. I just use macros, I don't understand them.

- To solve this problem, go back to Modeler and **Import** the FlatPlane.
- Use the **Rotate** tool (**Modify** menu or y) to turn it 180 degrees on the Y axis. Now the polygons will (surprise) be facing the right direction—toward the camera.
- Save it again and **Export** it back into Layout.

Well, the flat plane no longer turns inside out and the morph looks fairly good. However, look closely at the corners of the flat plane as it changes to the sphere. They fold back on themselves in a rather unpleasant manner. This is what prompted me to come up with...

Method Two

This approach uses the previous two objects and creates a third object, a cylinder, as an intermediary between the flat plane and the sphere.

- Load the FlatPlane object into Modeler.
- Click on the **Volume** button. It should say **Exclude**. If not, click on it again.
- Using the left mouse button, draw out a selection box in the Face view around the right half of the plane. The left edge of the selection box should be

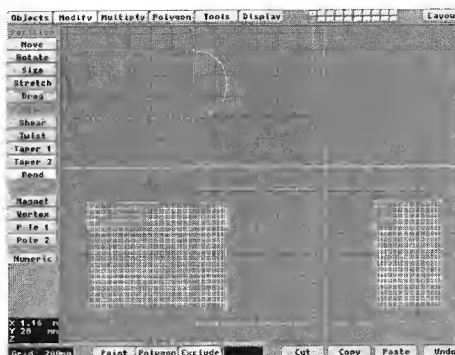


Figure 2: Use the Bend Tool to bend the right half of the plane back 180 degrees.

exactly at the center line (where $X = 0$). Now, when you go to bend this object, it will only affect the right half.

- Activate the **Bend** tool (**Modify** menu) and **Numeric**. In the requester, enter the following values:

Axis	X
Range	Automatic
Sense	positive (+)
Angle	180
Direction	90
Center	0, 0, 0

- Click on Apply.

You should see something like Figure 2. If the plane's end points do not bend back far enough where they are located at $X=0$, click **Undo**, select all of the middle points (where $X=0$) and use the **Set Val** tool (**Tools** menu) to set all of the middle points to $X=0$. Then you can repeat the above step. Sometimes, after centering an object, the middle points may be located at $X=-0$. Those points will not be affected in a volume that rests on them.

- Now turn off **Bend**, and move the volume selection box to surround the left half of the flat plane. This time, the right edge of the selection box should be at the center line. Choose **Bend** and **Numeric**, and use the same values as before, except change Sense to negative (-) and **Direction** to -90.
- Click on Apply.

You should have a cylinder. However, you'll notice that it's not centered.

- Use the Center macro to center the object at 0, 0, 0.
- Save this object as "Cylinder."

Setting Up the Morph

- Load all three objects—"FlatPlane," "Cylinder" and "Sphere"—into Layout.
- For Cylinder and Sphere, adjust the Object Dissolve to 100%.

- Set the **Metamorph Target** for Cylinder to Sphere, and the **Metamorph Target** for FlatPlane to Cylinder.
- Create a metamorph envelope (**E** button) for FlatPlane with keyframes at 0 and 30. Set the tension for keyframe 0 to -1 and the tension for keyframe 30 to +1.
- Make a similar Metamorph Envelope for Cylinder, but set the tension for keyframe 0 to +1 and the tension for keyframe 30 to -1.

Notice that the morph envelopes overlap. While the plane is morphing to the cylinder, the cylinder is morphing to the sphere.

- Make another wireframe preview and see what you think.

Because of the overlapping morph envelopes and the tension settings, the cylinder object is exerting more influence at the beginning of the morph, which makes for a more aesthetic change to a spherical shape. Note also that some of the points of the flat plane no longer travel in a straight line to their destination.

Some Morph Tips

- Any texture map that you have applied to the flat plane goes along for the ride when you morph it into the sphere.
- If you need to see the back of the sphere after the morph, do a one-frame dissolve to a sphere that has the points where the edges merged. Otherwise you'll see a seam.
- If you want to see the sphere morph down into a flat plane, simply set up your morphs so your plane object starts out 100% morphed and then goes down to 0%, or morph the sphere into the flat plane.

Morphing as Movement

Morphing is about movement. The points move. The polygons defined by those points move.

Why not use a morph to move a whole object?

Why would you want to, considering that LightWave allows you to move objects directly in Layout and set keyframes and all that?

Well, say your animation needed a mosquito. A rabid, hyperactive, directionally challenged mosquito, to be precise.

- Create a small (maybe 500 mm on each side) cube in Modeler.
- Move it (t) off to the lower righthand corner of the Face view.
- Save this object as "Box1."
- Move it again, this time over to the lower lefthand corner.
- Save as "Box2."

What's on the Disk?:

This month's disk includes scene files and objects from Joe Dox' Full Metal Hummer scene, a sample bones scene from Dan Ablan, and GVA's rotation and direction macro. Also included from last month's LWPRO is Gonzalo Garramuno's Hair Creation macro and Dan Ablan's Wood Scene.

Simple Space Stuff Part I

by Mojo

I know what you're thinking. You're saying to yourself, "What?! A feature in LWPRO on how to do space scenes? You've gotta be kidding me! Space is easy..."

Easy, huh?

Yeah, right.

Easy to get wrong, that is.

It may sound pompous and arrogant, but in the two years that *Babylon 5* has been in production I have yet to see any Toaster-generated space scenes that match the richness of Foundation Imaging's.

While I've seen plenty of space backgrounds behind many flying logos, they just don't measure up.

And it's not just me — many people have asked me questions about how we do it; questions that I honestly thought everyone knew the answers to.

So, no more assumptions — the following behind-the-scenes tour of *Babylon 5* will presume that everyone reading this needs to be taught, from the ground up, how to put together a convincing LightWave scene in outer space.

A Sky Full of Stars

The most important element of a space scene is the stars, which are those little pin pricks of light you see if you look up at the sky when it's dark.

Okay, I won't be that simplistic. However, I would suggest looking at the stars. You'll notice that there are a great deal of them (on a clear night, anyway). They also vary widely in intensity, as any number of them are closer, further, greater or smaller. This celestial variety must be copied within LightWave in order to be convincing. You'll need a large number of single-point polygons with varying surface names to accomplish this.

Two starfield objects are provided for you with LightWave—Stars is a random scattering of points with a single surface, making for a thin, flat starfield. ActualStars is a little better, since it contains realistic groupings of stars and several surface names that can be lightened or darkened for extra realism.



Figure 1: Overly bright lighting with fill has often been the norm for science fiction. However, don't look for anything like this to actually appear on *Babylon 5*.

Each of these on their own wouldn't satisfy even Duck Dodgers, but combined properly they create a starfield that would make Captain Picard proud.

On B5, we use two sets of each, for a total of four star objects, all rotated at different angles. The ActualStars object has had several extra surfaces added, all of which were painstakingly adjusted to get the desired effect (seven to 10 surface names created in Modeler should be enough).

Star densities were set with intensities anywhere from 10 to 100 percent. This was done by tweaking the **Surface Color** channel, not **Luminosity**, so some groupings could be given a slightly blue or reddish tinge. One of the Stars objects has also been given a slight dissolve to offset the full visibility of the other.

To avoid making all the object settings every time we do a space scene, we have a scene called 'Local Space' that contains only our starfield and nebula (more on that later). This way, after the main elements of a scene have been choreographed (like spaceships), all the space objects and their settings can be called up by simply hitting **Load From Scene** under the objects menu.

By selecting Local Space when requested, all our heavenly elements are put into place in no time.

With a little tinkering and just a few test renders, this formula should result in a starfield that looks as good or better than B5's in less than half an hour. That's only 2.8 percent of the time it took to create the entire universe.

Motion in the Ocean

Since most of you will probably not be satisfied with rendering a bunch of still frames, here are a few tips that might be handy when animating in space:

1. Always use **Particle Blur**, found in the Motion Blur section of the camera menu. This will streak your single-point polygons as the camera pans past them. It will look funny in stills, but when in motion it prevents the stars

from strobing and shimmering. The default setting of 50 percent for **Blur Length** should be fine for most scenes. Increase at your own risk.

2. Bring a power source. Electrical outlets are few and far between in our galaxy, and those who neglect this crucial element won't get very far. Battery supplies work well, but the new solar-powered cells on the market would be ideal when working in the direct sunlight of space.
3. Parent your starfields to a null object. Sometimes, to increase the feel of movement in a scene, we'll actually move the stars in an opposite direction from which a ship is traveling. Since we use four (and sometimes more) starfields, having them all attached to a null object can save minutes of tedious animating.

Hyperspace

At some point in their space career, every space cadet wants to copy the hyperspace effect from *Star Wars*, in which the viewer travels at such high velocity that the stars streak past like solid lines.

The idea is to have the camera move past many single-point polygons with a high degree of Particle Blur. This cannot be done with the normal starfield object, regardless of how fast you move, since all the polygons are located at the edges of this hollow object.

You'll need to make a solid block of particles using Modeler's **Point Distribution** macro. Try making several 1,000-point blocks, each with a different surface name and combining them into one (see reasons for various surface names above).

This particle object should most likely be far longer in the Z direction than X and Y, since you don't want to run out of stars as you travel down the center.

Load your starblock object into Layout and surface it appropriately. Upon your first render, you'll notice that the points conglomerate towards the center as the object recedes into the distance. This is easily fixed by adding fog to the scene or setting a distance dissolve for the object so only the desired number of stars can be seen at any given time.

As the stars approach the camera, they will slowly dissolve into view. By carefully balancing the Blur Length and the speed of the camera, you can easily mimic one of the more memorable scenes in science fiction film history.

Alternatively, for a more subtle effect, try loading several different starblock objects and move them past the camera at different speeds.

If you lower the Blur Length to a more manageable size, you can produce a statuesque zooming starfield, similar to the one seen on Star Trek or in B5's opening title sequence.

The B5 Blues

Perhaps the most prominent aspect of the Babylon 5 galaxy is the striking blue nebula, prevalent in the majority of scenes. While it is certainly pretty to look at, unbeknownst to most people, it serves a practical purpose which has made it essential to our shots.

In most cases, any major solar system would contain one sun, meaning that the majority of light radiates from one direction. Therefore, objects in space would be rich in shadow since only half of it can be lit at any given time. Most space-oriented productions add a lot of unnatural fill light to lessen these shadows and make their objects more visible. Let's face it—it is very difficult to see a ship shadowed in black against the black background of space. The fill light is a necessary evil.

Or is it?

Besides looking so nice, the blue nebula on B5 silhouettes the dark areas of ships in front of it, making their shape very clear to the viewer. This prevents an object from getting lost in a black background and therefore permits the celestial realism of high-contrast lighting without fill light. If it weren't for that nebula,

B5's special effects lighting wouldn't be very special (see accompanying images).

Although it is a simple object, many animators trying to emulate the look have been scratching their heads over the exact details of its creation. (Luckily, I'm in a good mood right now so I'll spill the beans but, please, don't tell anyone else.)

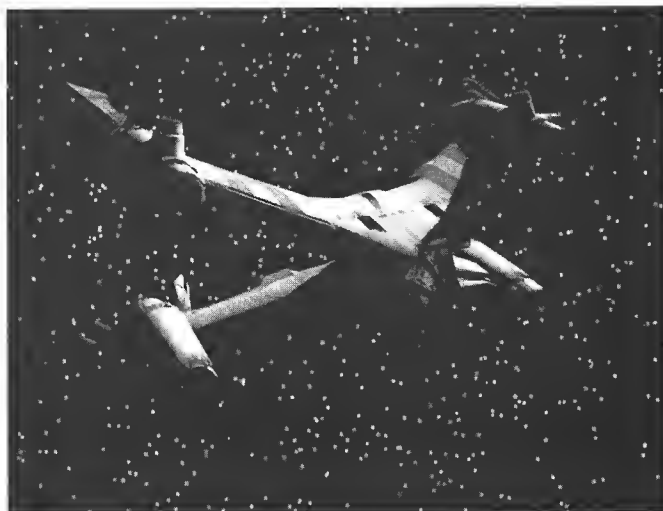


Figure 2: More realistic, single source lighting adds a dramatic touch to this shot. However, the dark areas of the model became lost against the black background of space.



Figure 3: The blue nebula silhouettes these dark areas, allowing for the instant recognition of the object's shape despite dark lighting.

The nebula itself was painted 1500x400 pixels on a PC using a paint program with sophisticated airbrush and blurring tools (sorry, Toaster Paint!). It needed to be very wide so the camera could pan across a scene without the nebula disappearing.

It also had to be mapped onto a curved polygon to hide its flat nature. This allows the nebula to stay within the camera shot, even with a pan of up to 180 degrees.

The painted image is actually a blue and black color map and the nebula object is sized larger than the starfields; otherwise, the stars behind the nebula

would be blocked out and the square edges of the object would be seen.

The nebula could have been painted in black and white and applied as a Transparency map to the object, solving some of these problems. Although this would also allow the color sliders to determine the color, Transparency takes considerably longer to render and so the color map path was chosen.

Different color nebulas are created by changing the actual image map; not the object. The nebula does not extend to the top and bottom of the screen as a purely stylistic choice. By not completely filling the screen, it prevents scenes from getting too blue, and allows us to tilt it on occasion for a different look (although the object can easily be stretched along the Y axis to fill more or less of the screen upon demand).

Of course, keeping the nebula to a minimum also reduces render times.

Nebulas Made Easy

Lazy animators who wish to create a nebula without even clicking a paint icon are in luck. A poor man's celestial phenomena is as easy as creating a simple sphere, sized slightly larger than LightWave's starfield objects.

Surface it with fractal noise of any color you like, with the noise size approximated at around 100,000.

A sphere is preferable to a flat plane, since it allows you to point the camera in any direction without losing the nebula. This produces nice results, although it will have none of the specific characteristics of the B5 nebula, like shape or hot spots.

The noise nebula also takes far more time to render.

Go ahead, be daring! Paint something! Even Dpaint AGA has what it takes to produce something usable. It's not difficult because the nebula is a fairly abstract shape. Start with a few bright splotches and begin to smear and blur them with various sized brushes.

Nothing described in this article is very difficult to produce. Hopefully, a few gaps

have been filled in for you or a few sparks have been ignited that will send you on your way to creating your own spectacular space scenes.

Next month, I'll dive a little deeper into the B5 universe and explain how more of it was created, including tips on how to build a vortex and make convincing laser beams.

LWP

Mojo is a Libra and therefore believes that all fish are pink and red ants wear tuxedos.

Simple Space Stuff

Part II—Cool Tricks From *Babylon 5*

by Mojo

Last month, I began explaining how the Babylon 5 universe was created. The mysteries behind majestic starfields and sweeping nebulas were finally revealed. This month, let's delve deeper into the secrets behind this LightWave-generated galaxy and see what else can be learned.

The Jumpgate

One of the most prominent effects seen regularly on the show is the hyper-space generator, a man-made jumpgate that creates a vortex allowing instantaneous travel to other galaxies. Fans of the show have raved about these sequences, although it is deceptively simple to duplicate.

The generators themselves are four simple objects that look somewhat like rails lined with solar energy cells. A single generator object was made and cloned three times, with all the pieces arranged into a diamond-like shape (all of them are parented to a null object to make the assembly easy to move).

The gateway comes to life with a violent explosion of lens flares that track down the length of the rails. Four flares (one for each generator) with identical envelopes meet at the far end and combine into one big flare, which appears to 'phase in' the vortex itself.

The vortex is a very long tube with a spiraling fractal noise pattern and transparent edge. The fiery noise pattern is orange and moves inward when a craft is entering hyperspace, or blue and moving outward when someone is leaving (the color difference is actually a visual Doppler shift which occurs when objects approach the speed of light—real science.).

The tube is just a long, subdivided object tapered at one end to force the perspective of it extending into infinity (we never see it at an extreme angle). The tube is subdivided so it can be twisted into a morph target which allows the fractal noise to appear as if it were spiraling. If the tube were straight, the noise would simply move in a straight line. But, if the noise-laden tube is morphed into a twisted

shape, the noise will follow these curves and travel in a spiral. Morphing is important, since the noise would not follow the contours of a twisted source object—textures always move in a straight line regardless of an object's shape. The object must be deformed after it is mapped.

A transparency map consisting of 95 percent black, fading to white only towards the end, is applied to the tube. This permits the tube to stay solid and feather off only at the edge.

The tube is sized to zero and dissolved out at the beginning of the sequence. When the four flares merge into one, the tube is dissolved in, sized up and moved out to the center of the generators. It is timed with the fading central flare to look as if it emerges from this point. Objects traveling down the tube are moved quickly and disappear with a lens flare when they reach the end. Since the tube is tapered, ships need to be sized down as they recede down the vortex in line with the forced perspective trick. It also makes them appear to be moving much faster.

Each one of these elements, separately, is very simple. But when tied together with the proper timing, a fantastic sequence is created. Keep this in mind when designing your own space scenes.

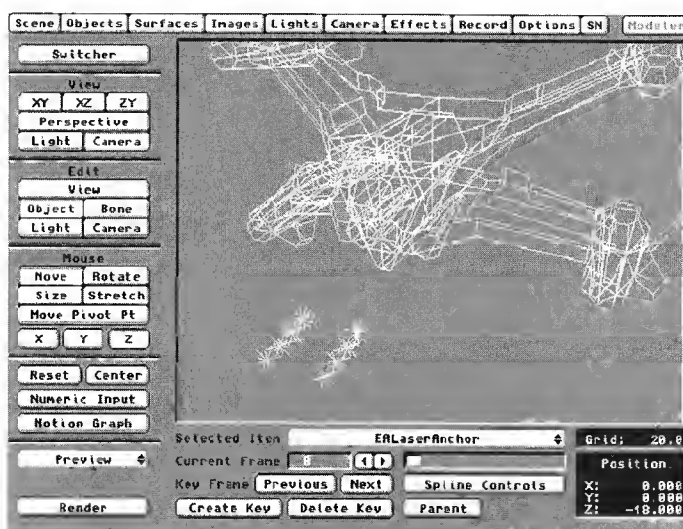


Figure 1: As this wireframe shows, the lens flares need to be positioned close to one another to avoid visible iterations.

Not Laser Beams

Space battles are an important part of science fiction and laser beams are essential for space battles. The laser beams used in B5's dogfight scenes went through several generations until we arrived at the satisfactory result we use now. [Note: In keeping with the show's strong footing in real physics and science, it should be mentioned that laser beams are not actually part of the B5 universe. The combat fighters are equipped with highly charged plasma weapons, which would result in 'bolts' being fired (a laser would be a solid beam) and a natural luminescence (lasers are invisible in a vacuum).] Early in production, we used thin, luminescent objects as plasma bolts which worked well, but I wanted to improve the effect.

For the episode "Signs and Portents" (which featured the first all-out battle sequence of the first season).

During the making of "And the Sky Full of Stars," there was a brief scene in which a Starfury fired its weapons at a Minbari cruiser. The firing was an afterthought and I didn't feel like taking the time to load the plasma bolt objects, so I simply threw a few lens flares into the scene. The angle was almost directly behind the fighter, so I didn't think anyone would notice that the bolts were round and not elongated.

Much to my surprise and delight, the final rendering revealed that the scene's motion blur also blurred the lens flares. In addition, since the flares were moving so fast, they were very blurred, resulting in an ovalic, elongated flare that looked much like a traditional, glowing, rotoscoped laser beam. When it came time for "Signs," I decided to experiment with this technique using lens flares as a permanent replacement for our plasma bolt objects.

Initial tests proved that at anything similar to a right angle (like a side view of a ship firing), the motion blur would stretch out the lens flares too much. This 'thinned out' the effect and the flare lost most of its concentration. Bolts would either have to fire slowly or high antialiasing (with more blur steps) would have to be used.

Neither option was attractive.

Instead, I tried tying several lens flares together in close proximity and this worked like a charm. They still blurred into a cohesive bolt, yet the extra flares filled in the 'thin' spaces created by motion blurring.

I created a scene with four flares for each bolt (the Starfury has two main guns), resulting in two rows of four flares each, all tied to a single null object. This null could then be permanently parented to the ship and moved along the Z axis (the Starfury's line of fire) to simulate bolts being fired (this hierarchy ensures that the bolts will always travel in the direction the ship is facing). The flares, of course, move along with the null (saving the headache of moving all of the flares individually).

Interactive lighting during these battle scenes is very important, so two of the lens flares that make up the bolts (one for each side) have their light intensity left on with a sharp falloff. This allows the plasma bolts to actually cast light onto surfaces they closely streak past. In addition, two lens flares (again, with light casting on and falloff) are attached to the end of the Starfury's guns. These flares are ramped up and down with each blast fired to simulate the blaze of the guns. They make the weapons seem much more powerful and the brief light cast upon the face of the ship as it fires is very dramatic.

For scenes in which plasma bolts are fired far from the camera, an image map of the blurred flares has been placed on a polygon (keyframing all the sets of bolts can be time consuming). This looks great from a distance, although to spice it up, a single, dissolved lens flare is placed on top of the polygon to give the beams a soft glow.

Creating these battle scenes is certainly a lot of work, but the results are worth it. This technique for creating plasma bolts (or, yes, even laser beams) may take a little time to master, but the brilliant glow of lens flares make this the only way I know of to achieve such classic results.

And the Sky Full of Clouds

Our first season featured an ambitious two-part saga, "A Voice in the Wilderness." This mini-movie featured more than one hundred new special effects shots, many of which took place in an atmosphere—something we had never tackled before.

One particular sequence had a shuttle blasting from below a planet's surface straight up into the sky. I had to create realistic looking clouds for this shot and knew that a flat polygon with a fractal noise transparency layer would be the answer.

However, as nice as this looks (the surface is included with LightWave), it wasn't quite good enough. The noise pat-

tern was too regular and there were just too many clouds in the sky. If there was a way to have a second layer of fractal transparency, I knew the effect would work better. But LightWave only has one transparency channel ... right?

Additive Transparency

Using LightWave's **Additive** feature (Surfaces panel), there is a way to fake a second transparency channel.

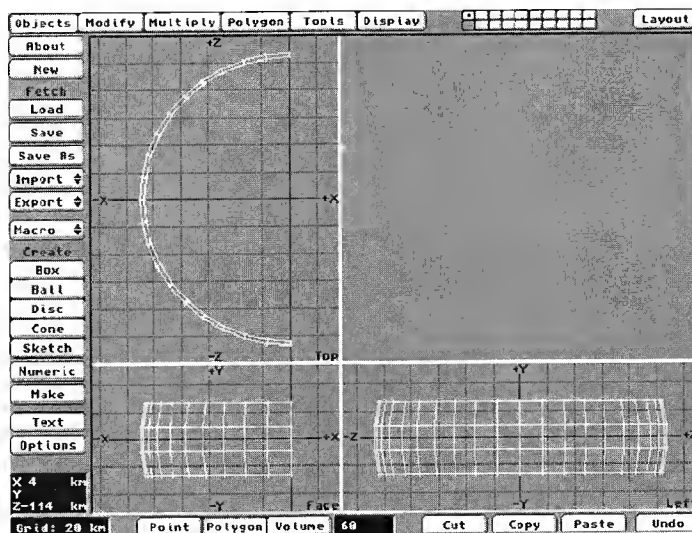


Figure 2: This image depicting the nebula object was mistakenly left out of Port 1. See last month's *LWPRO* for details.

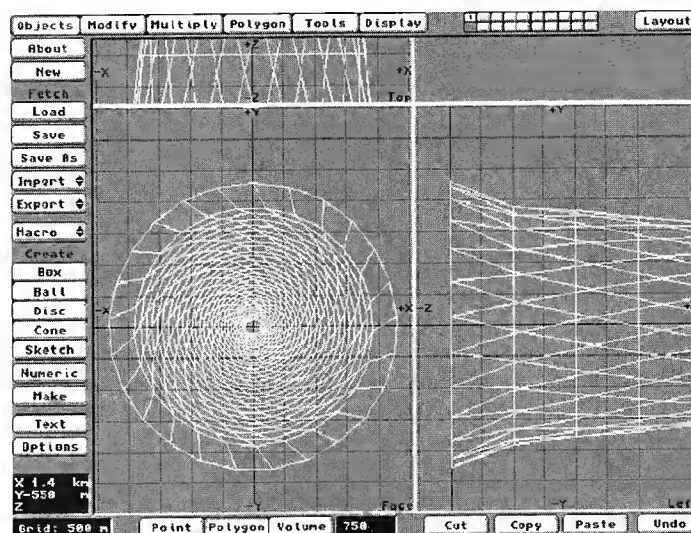


Figure 3: The morphed vortex object. Note that the polygons are triangular—tripling polygons meant to be deformed helps avoid rendering errors; three-sided polygons are more "flexible."

Surfaces that are additive allow parts of the scene that are behind them to be 'added' into the surface color of the surface. If a surface is 100,100,100 and the background is 100,100,100, the new surface color will be computed at 200,200,200. If an Additive surface has zero color, the background will be added completely in and the surface will appear transparent. By selecting **Additive** and by giving a surface 0 percent diffuse (so it appears black) and a

luminosity map which ranges from 100 percent to zero, you will effectively be giving that object transparency from 100 percent to zero. This technique works best on objects that would normally not be effected by light within the scene, such as distant clouds, smoke, electricity and other luminescent phenomena. To help you understand this trick, I'll show you how to create a lightbeam object without using a transparency map.

Make a one meter long lightbeam object out of a cone, making sure the tip of it is at 0,0,0 in Modeler and the base of it is located at one meter on the +Z axis. You can also delete the base polygon so the beam is "open-ended."

Back in Layout, surface it with whatever color you like, but instead of transparency, go into the Luminosity channel. Select the Grid texture and make the **Line Thickness** one (this leaves no spaces between grid squares and creates a solid block of texture) and the **Texture Value** 100 percent. Make sure you give the texture a **Texture Falloff** on its Z axis of 100 percent and set the Luminosity value under the surfaces menu to zero. This will make the object 100 percent luminous at its tip, falling off to no luminosity after one meter (the cone's end). Also make sure to set the Diffuse level to 0 percent. Click **Additive**, make sure the object's edges are **Transparent** and render a test. If done correctly, you should have a perfect lightbeam object without ever having touched the transparency channel. If you have render errors at the tip of the cone (showing up as facets even though smoothing is turned on), make your cone out of a one meter long disc and taper the tip so it is very small.

Of course, the transparency channel is still free. If this lightbeam were a car headlight, you could run a moving fractal noise texture through it and simulate fog or other debris. If you were unsatisfied with the type of lightbeam and wanted to use a custom transparency map, you could create the object in the traditional way and use the luminosity channel to add extra transparency.

Despite our Hollywood status, all the folks working on **B5** are still just a bunch of regular guys who love 3D. All of the unique effects we create are simply a result

of persistence and a little imagination. Despite all our expensive hardware, these are the elements that make our work special—elements that everyone reading this newsletter has in plentiful supply. There is no excuse why any of you cannot go forth and create your own spectacular 3D universes.

LightWave 101

Course 1A—Object Cleanup

by Taylor Kurosaki

Welcome to LightWave 101, where new animators with minimal experience will become more productive in LightWave. The more effective command of LightWave and Modeler's tools, the more time spent bringing your ideas to life and not wasting time merely navigating through the software.

With that in mind, the first step toward completing any animation is having an object or objects to animate. As important as modeling is, it's also essential to model sound objects and learn how to fix potential problems in models you create, as well as those you might acquire from other sources.

Improperly modeled objects, or objects which create rendering errors will detract from any animations you produce. The following techniques deal with the most common modeling problems and offer possible solutions. Most of the problems described won't necessarily be evident until the object is actually rendered. Importing from Layout to Modeler and exporting from Modeler back to Layout will be required to fix the actual iteration of the object from your given scene.

Remember to save your fixed object from Layout once it has been exported from Modeler. This is an important step to remember, as it ensures that you are always working with the same version of your object in both Layout and Modeler.

Merge Points

The first operation you should perform on an object you suspect has problems is **Merge Points**. From within the **Tools** menu, Merge Points can remedy one and two-point polygons (see below), as well as alert you to multiple polygons sharing the same space. (see Double Polygons below)

Merge points can also eliminate polygons with more than two points where multiple vertices share the same space. These polygons can render the same as one or two-point polygons but don't appear as such under the **Polygon Statistics** window. Performing a Merge Points exposes these problematic polygons for what they really are so you can remedy them.

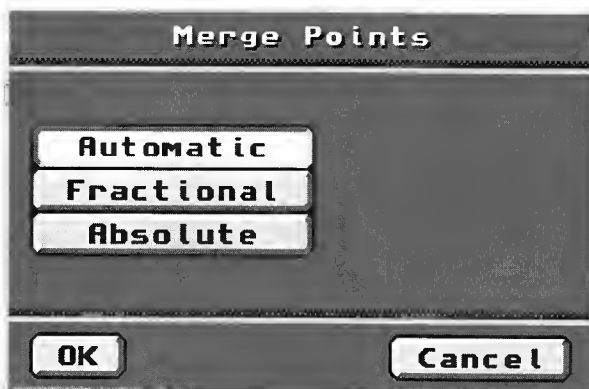


Figure 1: The Merge Points requester (m) allows you to merge points using a number a number of different formulas.

When you select **Merge (m)**, a requester with three options will appear: **Automatic**, **Fractional** and **Absolute** (Figure 1). In almost all cases, you will want to use Automatic as this merges only those points that share the same space. If you have two points that you know need merging but Automatic will not merge them, Absolute will merge points that are located within a user defined distance from each other.

Also make sure that you only merge points located in the same layer. You cannot merge points if they are in two different layers.

One-Point Polygons

One-point polygons render as dots or particles. Unless your model is a star field, plankton field or a cloud of particles, one-point polygons are probably a mistake. To search for one-point polygons, type the w key while the **Select Polygon** button (lower left screen) is selected to bring up the Polygon Statistics window. This window details the number of one, two, three, four, and greater than four vertex polygons in the current modeler layer. The statistics window also allows one to select polygons based on the number of vertices or surface names. Selecting the + key will select the chosen polygons, while selecting the - key will deselect them if they are selected.

Once the undesirable one-point polygons are selected, you can select **Cut** to remove them from your object. If the

point that made up the one-point polygon is also a part of another polygon, it will be maintained while the polygon is removed. If the point is not associated with any other polygons, it will be deleted along with its polygon.

Two-Point Polygons

Two-point polygons are similar to single point polygons in their problematic ways. Instead of dots protruding from your model, symptoms of unwanted two-point polygons are lines cutting through or laying on top of the object. The solution is the same as it is for single-point polygons.

Select all polygons with two vertices from within the Polygon Statistics window, then cut the polygon to remove it from the object. As with single-point polygons, if the points are needed by other polygons, they will be retained, otherwise they will be deleted along with the polygons.

Note: If your object intentionally contains one-point or two-point polygons, (i.e., the star field example above) LightWave version 3.5 now allows you to determine the rendered size of the **Particle/Line** within the **Objects** panel. Selecting **Small** renders particles the size of one pixel and lines one pixel thick, **Medium** renders 3x3 pixels, while **Large** renders particles 5x5 pixels in size and lines five pixels thick. Selecting **Automatic Particle/Line Size** renders a one pixel particle/line in Super Low Resolution and Low Resolution, three pixels in Medium Resolution, and five pixels in High Resolution and Print Resolution.

Flipped Polygons

Flipped polygons are a common modeling problem. A polygon with its surface normal facing the wrong way appears invisible when rendered. Think of a single-sided polygon as a two-way mirror; from the rear you can see through it, while from the front it reflects the surface values given back to the camera. The surface normal of a polygon in Modeler is the side that you will see when it is rendered.

To compound the problem of a flipped polygon, as you see through the back of the polygon nearest the camera, you will probably see through the back sides

of the polygons of the far side of the object. The net effect is seeing through the object to whatever is behind it. To fix a flipped polygon, either select it and hit the **Flip** button (**Polygon** menu) or the **f** key. You can also select the **Align** button and all polygons in the current layer will become aligned in the same direction. If you use **Align**, you need to check your polygons afterwards as they may all be facing inwards and you will need to perform a **Flip** on all polygons.

Note: For **Align** to work properly, you should perform a **Merge** points operation first.

Double Polygons

Double polygons are typically two polygons sharing the same set of points in the same space. Even if these two polygons use the same surface, rendering errors can still result. The polygons are essentially competing with each other for visibility, because their surface normals are facing the same way. In this case, hitting the **Unify** (**Polygon** menu) button will remove the extra polygon. If you have determined that there are, in fact, two (or more) polygons sharing the same space, yet **Unify** has no effect, chances are these polygons are using separate sets of points. In this case, perform a **Merge** points and then **Unify**. If merging points doesn't work, it is probably a situation where the two polygons in question are not directly on top of one another. If this is indeed what is happening, zoom in on the problem area and manually remove or cut the undesired polygon(s).

Non-planar Polygons

Non-planar polygons are those which consist of four or more points which are not located in the same plane. Due to their bent nature, these polygons cannot be rendered properly. From some angles, non-planar polygons might render fine, while from other angles they aren't even visible. When an object containing non-planar polygons is animated, the non-planars may flicker as the object moves in space. The best solution for non-planar polygons is to turn the four or more vertex polygons into triangular polygons.

Triangular, or three-vertex polygons, cannot, by their nature, be non-planar. Under the **Polygon Statistics** window (**w**), select the **+** button to the left of **Non-planar**, and then hit the **Triple** button from within the **Polygon** menu.

To be absolutely thorough in weeding out non-planar polygons, you must change the **Flatness Limit** to 0 percent in the **Data Options** window (**o**) within the **Objects** menu before selecting **Non-planars** and tripling them. Alternately, you may choose to **Triple** every polygon in your object, although this makes it more memory intensive and increase rendering times as well.



Figure 2: The Polygon Statistics requester contains information about all of the polygons in the current layer.

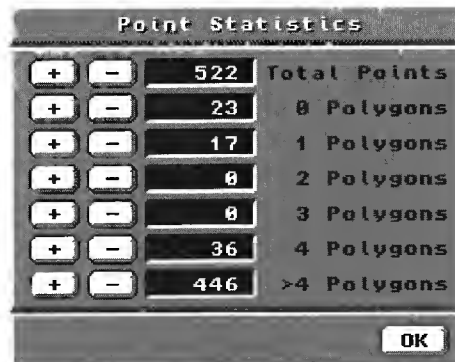


Figure 3: The Point Statistics requester gives you valuable point stats. You should not have any points containing zero or one polygon unless you are specifically making them.

Points with no Polygons

Periodically you may find an object which contains points with no polygons. Although individual points don't render, and therefore don't cause rendering

errors, it's still a good idea to eliminate of these extraneous points, keeping your object clean and simple in the event you want to go back and modify it later.

To find these rogue points, click on the **Point** selection button, press **w** to bring up the **Point Statistics** window, select all points which are associated with 0 Polygons, and **Cut**.

Last Resort: Double Side

If you encounter the types of problems described under non-planar or flipped polygons, yet none of the solutions seem to work, or if the object you are working with looks to have seams or holes in it, and you can't figure out how to fix it, there is one final potential solution. Copy all the polygons of your object, or at least all of the potentially problematic polygons, **Flip** the originally selected polygons, **Paste** the copied polygons back down, and **Merge** points.

You have given each polygon a duplicate with an opposite facing surface of the same name. This definitely fixes flipped polygons, and it may help with some cases of non-planar. Alternately, you can select **Double Sided** under the **Surfaces** panel in **Layout**.

Obviously, it would be preferable to actually fix the problems in your object, as opposed to merely covering them up. However, some objects can seem extremely complicated, especially if you aren't the original modeler. If you can't fix it, at least make it appear well done.

Hopefully these scenarios will be helpful when you invariably experience problems rendering objects. Ideally, when it comes to your own **LightWave** models, you will use these methods to fix problems as you are building. This is the least painful way to work, and if you're conscientious and thorough, problems can be avoided by the time you are working in **Layout**.

Next month, we will discuss different strategies and approaches to modeling. The use of simple shapes to construct more complex objects will be covered, including the use of primitives, lathing, cloning and subdividing.

LWP

Taylor Kurosaki is the former Digital Systems coordinator for Amblin Imaging and seaQuest DSV. He now devotes all of his time to animating so he can miss all those boring meetings and turn Pearl Jam up while he works.

EDITOR'S MESSAGE

continued from page 3

Finally, **LightWave** will gain the full acceptance it deserves. Many people scoff at the fact that **LightWave** runs on an Amiga. They assume it can't be that good, but when they are shown a series of quality **LightWave** animations, they are most often surprised. **LightWave** has had to fight for acceptance at many levels, and the move to these platforms will help validate it as the powerhouse it is.

The great part about this is that **LightWave**, in order to be as good as it is on an Amiga, is very well opti-

mized. It is finely tuned to perform, and the continuation of this process by Stuart Ferguson and Allen Hastings will ensure that **LightWave**, on any platform, will maintain its excellent position.

What new features can you expect? For one, inverse kinematics (IK), which will allow you to pull on a finger and have the arm follow, plus expanded plug-in textures and new post processing plug-ins, just to name a few.

Along with **LightWave 4.0** will come a new manual

written by yours truly. Since NewTek has commissioned me to write the manual, I expect to be quite busy over the next few months and may have to rely on some guest editors to get **LWPRO** to you on time (now you know why this issue is late!).

John Gross
Editor

LightWave 101

Course 1B: Basic Modeling Strategies

by Taylor Kurosaki

This installment of LightWave 101 will focus on modeling techniques which allow you to create seemingly complex objects simply and easily. Just because you don't have a tremendous amount of modeling experience, or aren't nicknamed "spline master" doesn't mean you can't model high-tech, cool-looking objects in LightWave.

The most important thing to have is the ability to visually break down any given object into its most basic components, and then have enough knowledge of Modeler to construct these basic shapes. Pieces which do not resemble any of Modeler's primitives can usually be created simply with any number of Modeler's tools. Finally, these basic pieces are assembled like "Legos" to create your final shape.

Note: To successfully complete your object it is essential to view your work in progress. I strongly suggest that you Export your object from Modeler into Layout and render tests from various angles. Subsequently your object(s) should then be saved from Layout, and as part of a default scene for your model. To make modifications, Import the object back into Modeler and edit the imported version. This method ensures that you are always working on the most recent version of the object. It also keeps surface values intact in Layout.

Sputnik: The Easy Way

A satellite is the perfect example of a model which can look fairly impressive without a great deal of frustration. For the purpose of this tutorial I will use the example of the satellite pictured in this issue and included on the LWPRO disk. The following examples are merely starting points to get you thinking in the right direction with regard to objects that you design. You might want to experiment by modeling a similar satellite-based on some of the principles mentioned within, and later apply these methods to other objects.

Two pieces of advice: 1) Never be afraid to experiment—there is no universally correct way to build LightWave objects. 2) Modeler is a production tool, not a conceptual tool. Paper and pencil are a much better way to figure out what it is you want to build. Sketch before turning on your computer.

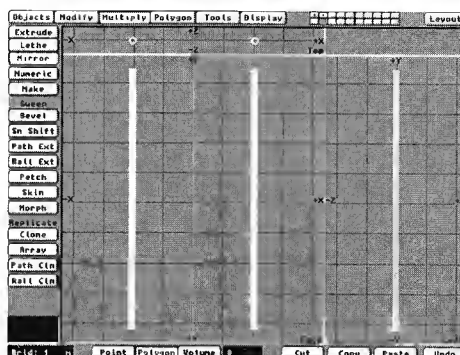


Figure 1

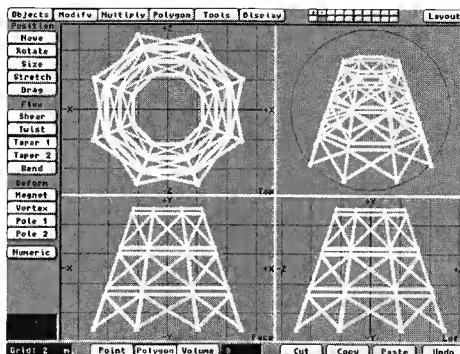


Figure 3

Primitives

Effective use of primitives is essential to successful modeling. The most obvious use of primitives in the satellite model are the upper and lower girder sections. These sections were built by creating a long, narrow Disc along the Y-axis for one of the support beams and Mirroring it across the Z-axis (Figure 1).

Next, another disc, with a slightly smaller diameter, is positioned horizontally between the two support beams. To create a diagonal crossbeam, **Copy** the horizontal disc and **Paste** it into another layer of Modeler.

Now **Rotate** (Modify menu) the copy by about 30 degrees on the Z-axis, and **Stretch** (Modify menu) as necessary to make the ends intersect with the support beams.

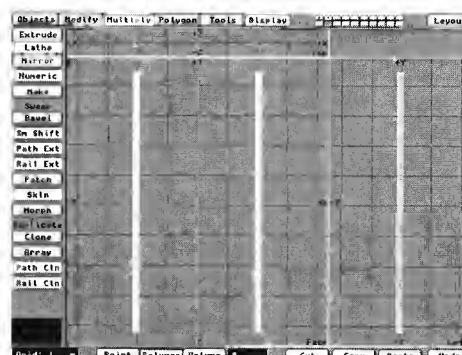


Figure 2

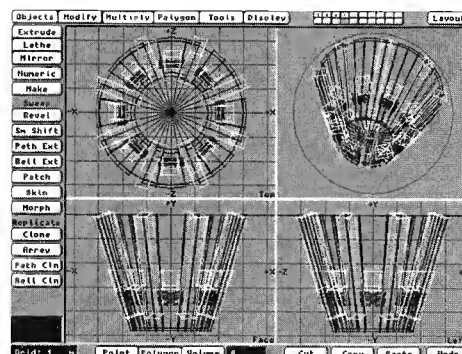


Figure 4

Mirror (Multiply menu) the copy on the Z-axis to create the opposite diagonal beam. **Copy** and **Paste** the diagonal and horizontal beams twice and five times, respectively, into various layers and **Move** (Modify menu) the copies along the Y-axis to complete the first girder section. (Figure 2)

Next, select and **Cut** the Polygons which make up one of the vertical support beams, so the original support beam and all of the cross beams are left. (one-eighth the final girder structure). Finally, **Clone** (under the Multiply menu) the girder section seven times around the Y-axis at an angle of 45 degrees to create the rest of the object.

Finally, use **Taper 1** (Modify menu) centered on the X and Z axis to obtain your final shape. (Figure 3)

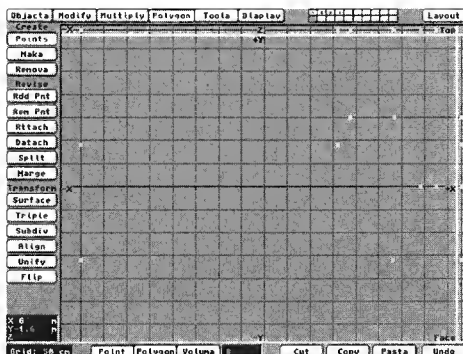


Figure 5

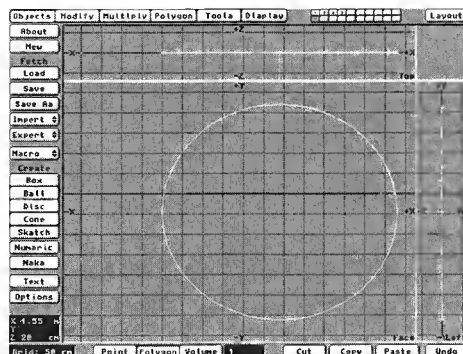


Figure 6

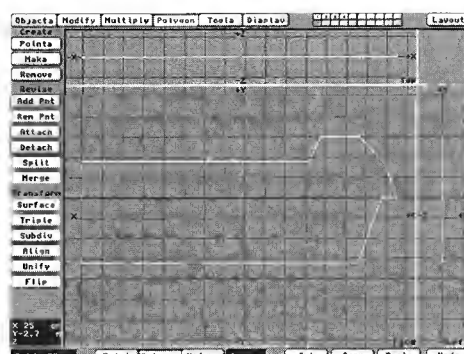


Figure 7

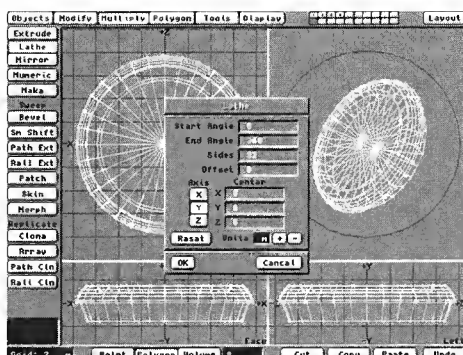


Figure 8

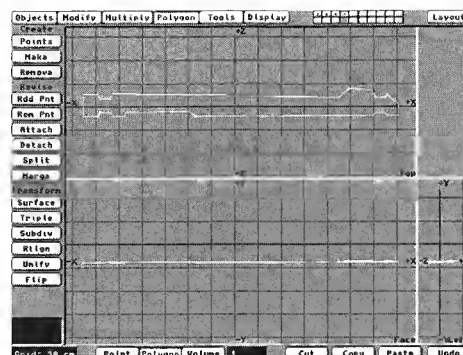


Figure 9

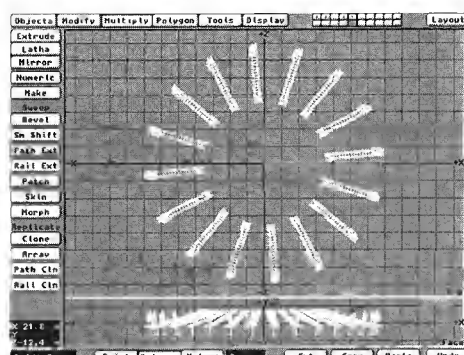


Figure 10

Other uses of primitives on the satellite are found in the lower, center section. This is where boxes are sized and positioned to intersect with the main body and are cloned around the Y-axis to give the satellite some detail and dimension. (Foreground layer-Figure 4)

Sculpt and Lathe

The satellite's saucer section was easily created with a method I call "sculpt and lathe." In this instance, "sculpt" refers to the process of placing Points (Create Points under the Polygon menu) to sculpt the cross-section of the shape you want to build. (Figure 5)

In the case of the satellite, the outer ring of the saucer section is comprised, partially, of an arc. This shape is difficult to reproduce merely by placing points; it is much more effective to utilize primitives once again.

Select **Disc** (Objects menu) and draw out a Disc that intersects the two Points between which you want the arc. Once you **Make** (Enter) the Disc, select it (Figure 6) and **Remove** (k) the polygon which comprises it.

Now select and **Cut** (x) the points of the disc which do not fall between your original outer points. You are left with a perfect arc section for the satellite. To ensure a smooth arc, you must have an adequate number of points. Therefore before Making the Disc, select **Numeric** (n) and increase the number of sides to at least 32 or even 48.

Now that you have the outline for the saucer-section's shape, select the Points in order and **Make** them into a polygon (p) (Figure 7). To complete the shape, select the cross-section polygon and **Lathe** (Multiply menu) it around the Y-axis. (Figure 8)

Bevel, Julianne and Bend

The final components of the satellite are the reflector arms. The basic shape is made by creating points and making a polygon (Figure 9) in much the same way as the cross-section for the saucer piece.

To give the reflector thickness and an interesting edge shape you might want to **Bevel** (b) it with an inset of -140mm and a shift of -40mm (assuming your reflector is as big as mine). Keep in mind, however, to retain a top on the reflector and enclose your final object, **Copy** the original polygon and **Paste** the copy into another layer before performing the **Bevel** operation. Upon completion of the bevel, **Cut** the copy of the original polygon, **Paste** it into the same layer as the beveled polygon, and perform a **Merge** Points (Tools menu).

It's also a good idea to check the surface normals of all polygons and **Align** and **Flip** (Polygon menu) if needed so all surface normals are facing out.

To give the reflector arm a nice bend, perform the **Julienne** Macro (Objects menu) along the X-axis with anywhere from 10 to 20 divisions. Once it's been sliced, use the **Bend** tool (Modify menu) to arch the reflector arm slightly along the X-Axis. Enter the fol-

lowing parameters in the Bend Numeric requester:

Angle: 11 degrees

Direction: 0 degrees

Center: 0, -0.45, .15m

All other parameters should be left at their default settings.

Finally, **Clone** the finished reflector fifteen times at a rotation of 22.5 degrees around the Y-Axis to produce the rest of the reflector arms. (Figure 10)

What Now?

The remainder of the satellite object was modeled using variations of the techniques described throughout this tutorial. Employ the strategies above to complete an object of your own, or try to construct a satellite similar to the example.

Next month LightWave 101 will focus on surfacing strategies within Modeler and mapping techniques in Layout. Until then, keep your feet on the ground and your cursor close to the Save button.

LVP

Taylor Kurosaki is a CGI Artist at Steven Spielberg's Amblin Imaging. While not writing LightWave 101 he works as a member of the visual effects team on NBC's seaQuest DSV. Questions, comments, suggestions and ironing tips should be sent to his attention at 100 Universal City Plaza Bldg. 447, Universal City, CA 91608.

LightWave 101

Course 1C: Basic Surface Strategies

by Taylor Kurosaki

This installment of "LightWave 101" will begin to explore one of the most critical components of 3D: object surfacing. In terms of creating a photorealistic CGI environment, effective surfacing is crucial. For example, there is currently a completely computer-generated commercial for Levi's Jeans featuring a figurine pulling rubber balls out of a shoebox.

To be sure, the modeling involved was extremely straight-forward and simple, consisting only of basic variations of spheres, boxes and cylinders. The animation was fairly basic and the movements weren't exactly lifelike (nor did they necessarily have to be). Even the lighting was less than realistic with its hard-edged, ray-traced shadows and lack of natural reflective light.

That said, I am still constantly fascinated with this commercial. The surfacing work in this spot is so realistic that at times the distinction between computer generated and reality is blurred. It isn't even the texture maps themselves which are excellent; in fact, the only texture maps I can remember are the stripe and star on one of the rubber balls, and the wood texture on the figurine. What makes the surfacing so realistic, in this case, is the way in which light affects the objects, the way light would affect similar objects in reality. The rubber ball appears to "feel" rubber. The figurine looks to be made of stained and painted wood. The surfaces on these objects make them seem tangible.

The satellite object from last month's issue will again be used here as an example. Obviously, I want the satellite to look realistic, but more specifically, it should look like it's been in orbit for quite some time. It should appear metallic yet grungy, not unlike the "dirty" ships in the Star Wars movies. This can be accomplished easier than you may think.

The areas of the satellite we will surface are the Upper and Lower Girder sections, the Ring section and the Center structure. Let's begin by first selecting



Figure 1

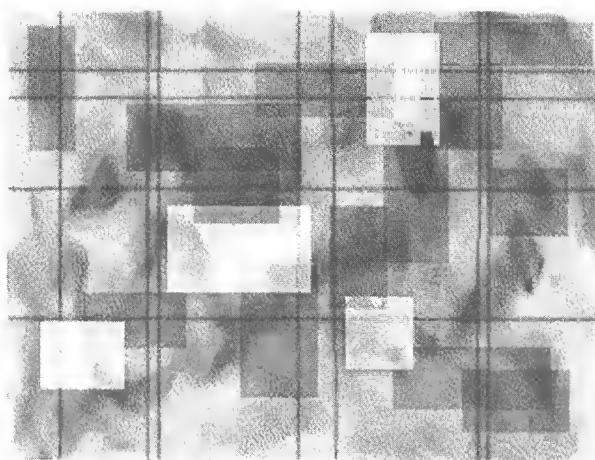


Figure 2

the Surface Color for these three sections. Keep in mind that this is merely a starting point, as the Diffuse percentage will greatly affect the appearance of the surface color.

For the Ring section I have chosen the values of 200, 210, 225, which is a slightly bluish, light gray. To create contrast the Center section uses the darker val-

ues of 100, 110, 120. The Upper and Lower Girder sections use a dark gray of 60, 60, 60. The resulting surfaces appear in Figure 1.

Obtaining a dirty metal appearance involves primarily a combination of Specular and Diffuse image mapping. Before diving into the theory behind this surfacing technique, a few definitions are necessary: Diffuse is the amount of light returned from the object back to the camera. In the real world, an object which is 100 percent diffuse returns all light hitting it, while an object which is 0 percent diffuse absorbs all light reaching it (thus making it appear black). Although LightWave doesn't actually reproduce this phenomenon (it's called radiosity), it does simulate it by reducing the color values by the percentage of the diffusion. For example, an object with a Surface Color of 255,0,0 with a light of 100 percent intensity perpendicular to it and a Diffuse value of 100 percent will appear as pure red when rendered. Reduce the Diffuse level to 50 percent, and the rendered color is equivalent to 127,0,0. Lower the Diffuse to 0 percent and the object appears black (0,0,0).

Specularity refers to the amount of shine an object appears to have. A polished chrome surface would be extremely specular (100 percent), while dry clay would have no specularity whatsoever (0 percent). This tutorial will describe how to use a combination of Diffuse and Specular attributes to simulate a dirty, weathered, metallic surface.

Diffuse Mapping

To obtain a tiled, metal plate surface, a map similar to the one shown in Figure 2 works well. It can be painted in grayscale in any paint program by selecting rectangular areas and filling them in with varying shades of gray.

My preferred method of painting this type of map is to use Adobe Photoshop, beginning with a light-gray

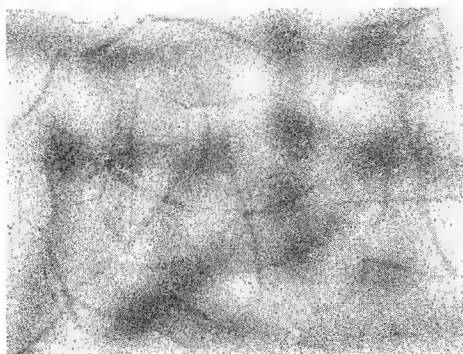


Figure 3

background. Independently select various rectangular areas, and adjust the contrast and brightness of the selected areas into differing regions of light and dark. Finally, perform a noise or blur operation on the whole image to grunge it up. The darker areas of the map will correspond to areas of less diffusion, while lighter areas appear more diffuse.

More specifically, totally white areas (255,255,255) translate to areas of 100 percent diffusion, completely black areas (0,0,0) result in areas of 0 percent diffusion, and medium gray areas (127,127,127) render with 50 percent diffusion. The end result is a light and dark pattern on the object which consists of several shades of the object's original Surface Color.

Due to this fact, Diffuse Maps (and other surface maps, except Surface Color) need only be grayscale images, as their color information is discarded in favor of the object's Surface Color. The image in Figure 2 was used as a diffuse map on both the Ring and Center sections of the satellite with the following settings:

Ring Section

Texture Type Cylindrical Image Map
Pixel Blending On
Texture Axis Y Axis
Texture Size 3.75, 16.0, 3.75
Antialiasing On
Width Wrap Amount 7.0

Center Section

Texture Type Cylindrical Image Map
Pixel Blending On
Texture Axis Y Axis
Texture Size 6.26, 8.0, 6.26
Antialiasing On

All other values should be left at their defaults.

For the satellite's Upper and Lower Girder sections, the map in Figure 3 was painted. This map was also created in Photoshop using the airbrush tool with noise added, although it can be created just as easily in any paint program. When applied as a Diffuse map, the darker areas appear as dirty areas on the surface, while the lighter areas appear clean, allowing the true



Figure 4

surface color to show through more visibly. The following values were used here:

Upper and Lower Girder Sections

Texture Type Cylindrical Image Map
Pixel Blending On
Texture Axis Y Axis
Texture Size 10.55, 50.0, 10.55
Antialiasing On

Again, any values not listed were left at default.

Figure 4 shows the Ring, Center, and Upper Girder sections of the satellite with their Diffuse maps in place.

Specular Mapping

The satellite is now comprised of light and dark patterns through the use of Diffuse maps. On the Ring and Center sections, the Diffuse maps appear as weathered metal plating, giving the satellite a mechanical, assembled look. The Diffuse map on the Girder sections serves a different purpose—to look like dirty piping. The satellite is now shaded like it is comprised of metal, but it doesn't have the shiny properties of metal.

This is where the Specular Level setting is helpful. If the satellite was perfectly clean and polished, it would have a constant specularity across its surface (somewhere between five and 20 percent, for example). In our case, however, the Specular level would not be constant due to its tarnished state. Dirty areas would appear less specular than cleaner areas.

This is the perfect application for a Specular Image Map. In the case of the satellite's Girders, the shape of the Specular Map should be the same as the Diffuse Map, due to the fact that the dirty (less diffuse) areas should correspond to less shiny (less specular) areas.

Therefore, a variation of the Girder's Diffuse map will work well as its Specular map. In this case I darkened the Girder's Diffuse map, saved it, and applied it as a specular map (Figure 5).

Darkening will lower the overall specular level throughout the map. [Note: Surfaces that are too specular are always a sure giveaway of CGI. A little specular goes a long way, so keep your specular maps on the darkish side.]

I then applied the same specular map to the Ring

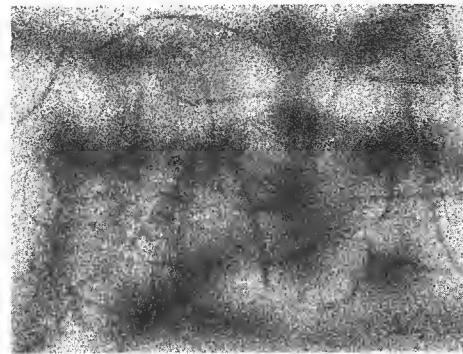


Figure 5

section of the satellite. This map will add the highlights necessary to complete the metallic look for this object. See the color section for the completed satellite with these surfaces in place. The values used are listed below:

Upper and Lower Girder Sections

Texture Type Cylindrical Image Map
Pixel Blending On
Texture Axis Y-Axis
Texture Size 10.55, 50.0, 10.55
Antialiasing On

Ring Section

Texture Type Cylindrical Image Map
Pixel Blending On
Texture Axis Y-Axis
Texture Size 16.05, 103.75, 16.05
Antialiasing On
Width Wrap Amount 5 0.10

Other Maps

Another type of map which adds realism to your objects is a Bump Map. This type of map will cause bumps and ridges to appear in the surface of the object, depending on the shape of the image. With a Bump Map, the dark areas of the image appear to be inset from the surface, causing shading to "fall into" those areas. With the satellite object, use the diffuse map image and settings as a Bump Map on the Ring and Center sections. A Texture Amplitude anywhere from 100 percent to 200 percent should work well.

For a more general overview of the Surfaces panel, read Grant Boucher's "Intro to Surfaces" article in the February 1994 issue of LWPRO, or see your Toaster manual. See you next month in "LightWave 101."

LWP

Taylor Kurosaki is an animator at Amblin Imaging. As usual, the majority of the techniques described above were systematically, though painlessly, taken directly from the brain of Eric Barba, among others. Thanks guys. Questions, comments and a comb should be sent to Taylor's attention at 100 Universal City Plaza, Bldg. 447, Universal City, CA 91608.

LightWave 101

Course 1D: Animating with Splines

by Taylor Kurosaki

To animate is to bring to life. However, creating life is often an incredibly painstaking process. Just imagine the work involved in hand painting thousands upon thousands of cells for an animated film. Now imagine a cartoon animator only having to paint every other frame of an animation. How about every fifth frame? Every tenth? Every hundredth? The labor and time saved would be enormous. Now apply this principle to computer animation. Imagine how tedious it would be to keyframe the camera, every object, and every light at every frame of your scene. Thankfully, this is not the case. The computer calculates the positions of the components of the scene based on their respective keyframes immediately preceding and following the current frame. In LightWave, interpolation, as this mechanism is called, comes to us in the form of splines. Certain characteristics attributed to keyframes affect the spline and determine how the object's position is interpolated. In layman's terms, animating with splines in LightWave is not unlike having your own overseas animation department.

This installment of "LightWave 101" will cover the use of splines in LightWave animations. More specifically, effective use of the **Motion Graph** and **Spline Controls** will be illustrated, along with the **Align to Path** feature.

A few words of general advice: add keyframes as discriminantly and strategically as possible. Don't get me wrong, I'm not suggesting skimping on keyframes in any way. Rather, be wary of ending up with a messy animation full of unnecessary keyframes. This usually occurs when an animator tries to fix a problem in an animation by adding extra keyframes rather than first trying to adjust those which already exist. With very complex animations involving multiple, interacting objects, it may seem easier to "tweak" a motion by adding keyframes to counteract the negative effects of others rather than altering existing keyframes. This only serves to further complicate the animation when you invariably must go back to make later changes. Get enough of this and you may end up with a monitor lying in pieces on the sidewalk below. Don't be seduced by the dark side...fix it right the first time.

The Motion Graph

The Motion Graph window is an incredibly helpful feature that debuted with the 3.0 release of LightWave. It provides LightWave animators with a visual representation of their animations and allows interactive adjustment of an object's, camera's, light's, or bone's movement. The main area of the window contains a graph which plots movement along the three axes, heading, pitch and bank angles, object scale along any axis, and velocity. Values for the various channels are plotted vertically, while time is plotted horizontally. Through this interface, one can **Add**, **Delete**, **Drag**, **Shift** and **Scale** keyframes, and observe visually the results of changes made. While you can do all of this in the main Layout window, it is often easier to use the motion graph controls. Select the motion graph for an object, bone, light or camera by selecting the item in Layout and clicking on the **Motion Graph** button or by pressing the "m" key any time you have an item selected. Remember that dragging a keyframe with the left mouse button allows you to move the key up or down (except while in the Velocity channel) and dragging with the right mouse button allows you to move a keyframe right or left.

Spline Controls

One of the most useful purposes of the Motion Graph is the ability to visualize the effects of **Spline Controls** on a motion path. The Spline Controls available in LightWave are **Tension**, **Continuity**, **Bias** and **Linear**. Valid settings for the first three controls are -1 to 1, with 0 as the default. Linear causes an object to move directly from the previous keyframe directly to the selected keyframe without any spline deviation. When Linear is selected, it overrides all other spline controls from the keyframe on which it is selected to the keyframe immediately preceding it. It's important to note that spline controls are still valid to use at a keyframe with Linear selected since they can be useful in the path leading to the next keyframe.

Tension controls the velocity at which an object travels through a keyframe. A negative tension setting will cause an object to accelerate through the

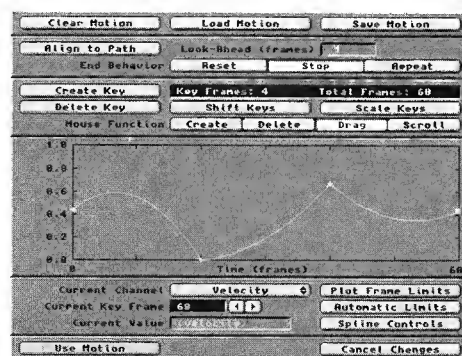


Figure 1

keyframe, while a positive tension setting will cause an object to slow as it passes through the keyframe, then accelerate gradually away from it. A tension setting of 1.0 will cause an item to come to a momentary stop at a keyframe. Figure 1 shows the effects of tension on the Velocity of an object. Frame 20 contains a tension of 1, while frame 40 has a tension of -1.

Continuity affects the smoothness at which the keyframe is incorporated into the spline path. With a continuity setting of 0, the spline curves fluidly through the keyframe. With continuity set to an extreme -1, the spline abruptly changes course directly at the keyframe, causing an object to alter its direction immediately as it reaches the keyframe. A continuity setting of +1 causes the spline to over-compensate before and after the keyframe and "stutter" at the keyframe. Positive continuity values are generally not common, unless continuity is used in combination with another spline control.

Bias allows one to offset the spline curvature before or after the keyframe. With a positive setting, the object will overshoot the keyframe, while a negative setting will result in a spline path which anticipates the keyframe.

All these Spline Controls can be altered directly in the Motion Graph using the keyboard. Hold the t, c, or b key and click and drag right or left to adjust the spline by eye, or enter the values numerically by clicking the **Spline Controls** button in the Motion Graph or from within Layout.

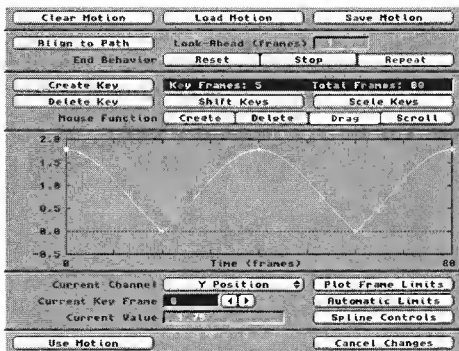


Figure 2

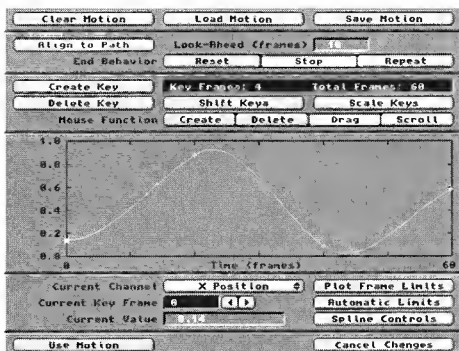


Figure 3

Follow the Bouncing Ball

Using Spline Controls is an excellent way to easily animate a bouncing ball. Figure 2 shows the spline path for the Y Position of a bouncing basketball. To reproduce the way the ball slows as it reaches its apex, surrendering to gravity, use a Tension setting of 1 for the keyframe at the top of the bounce (Frames 0, 40, 80; Figure 2). A tension setting of 1 literally causes the ball to stop briefly (for the duration of one frame, to be exact) before heading earthward. Conversely, to accentuate the force of the ball slamming solidly to the ground, use a continuity setting of -1 (Frames 20, 60; Figure 2). To keep the ball continually bouncing select an End Behavior of **Repeat** from the Motion Graph. In order to make a smooth loop, just make sure the first and last keyframes have identical values. Keep in mind, though, that a ball would not endlessly bounce unless someone were dribbling it, and upon each bounce, the ball would not rebound as high.

Driving School

Bias is a useful way to simulate an object with a lot of velocity and mass. Examples which come to mind are a speedboat or a race car. In these examples, the mass of the object, combined with its forward velocity, causes the object to overshoot the keyframe as it begins to turn. Use a bias setting of 1 to offset the

curve in the spline path to after the keyframe. Figure 3 shows the X position of a race car as it drifts around corners, or slides on an icy road. Notice how the spline remains nearly straight until after the keyframes at frames 20 and 40. Whereas positive bias values would be used for a skidding race car, negative bias values would be used to simulate a perfectly in-control car. With a bias of -1, the curve in the spline path occurs before the keyframe as the car sets up and completes its turn before the keyframe (Figure 4). In conjunction with bias, **Align To Path** can be used to keep the car's heading aligned with its direction of travel. In the case of the skidding car, a **Look-Ahead** setting of five to 10 frames accentuates the sliding motion. For the in-control car, use a look-ahead value of 1 frame. Note: When using **Align To Path**, the object will reset its heading when it reaches the last keyframe, due to the fact that it has no more frames to look-ahead to. If you wish to use **Align To Path** and you experience this "last frame resetting," you may wish to extend the motion path past the last frame you will see in order to keep your object aligned.

Terminal Velocity

The **Velocity** channel is an especially useful component of the Motion Graph. It is an effective way to

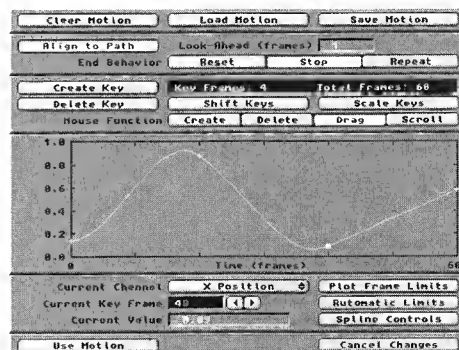


Figure 4

diagnose and smooth jerky animations in a number of ways. Assuming you are trying to produce an animation with a fairly constant velocity, the motion graph in Figure 5 would reveal some significant problems. Dragging the mouse while holding the right mouse button, the spline can be smoothed by spacing keys farther apart to lower the velocity or bringing keys closer together to raise the velocity. Alternately, tension can be used to raise or lower velocity values to smooth the spline. Figure 6 shows the path from Figure 5 smoothed out by adjusting tension and sliding keyframes. Ideally, however, inconsistent velocity values can be averted altogether by establishing a basic velocity by creating the first and last keyframes

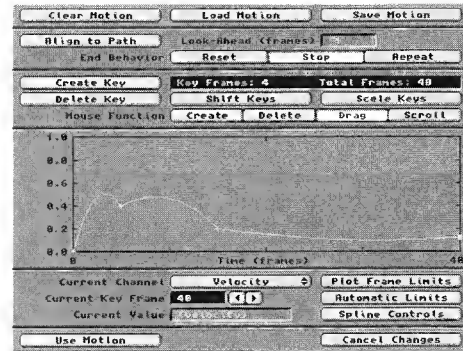


Figure 5

of your animation first. Keyframes in-between are then created subsequently to conform somewhat to the original two-keyframe spline. This is easy to do by going to the frame you wish and moving the item slightly from its original path. The resulting final

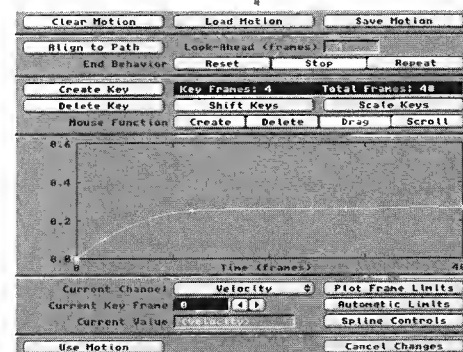


Figure 6

velocity stays in the neighborhood of the previously set values.

While these are very rudimentary examples of spline control use, the principles remain true as you tackle more complicated animations. Character animating, in particular, can be a very frustrating task. The Motion Graph becomes indispensable as it allows you to smooth out jerky motions visually and interactively. In general, however, spline controls enable you to create complex motions with minimal key-framing. Not only does this save time, but it keeps animations streamlined and simple. When you find yourself working on an animation containing several hundred keyframes, you gain a new level of appreciation for this fact. Truth be told, shattered CRT tubes are a pain to clean up.

LWP

Taylor Kurosaki is a visual effects artist at Amblin Imaging. Topic suggestions and comments should be sent to his attention at 100 Universal City Plaza, Bldg. 447, Universal City, CA 91608, or to John Gross on-line.

Corrections:

The Water Surface image in the July issue was credited to John F.K. Parenteau. It was actually done by Greg Teegarden. In the September Issue, image descriptions were mixed up in "Metaform Basics" for images 6, 7, 8 and 9.

Narn Cruiser

The Damaged cruiser from Babylon 5.

See page 12



Robo Explode

Lens flare clusters and fireball object make an explosive combination.

See page 14



Humanoid

In a complex setting, we must take into account how light falls on our environment.

See page 6



Xevious

The Essence attribute Xevious was loaded to create this image using the RAD machinery texture

See page 22

